DVFS using clock scheduling for Multicore Systems-on-Chip and Networks-on-Chip

POLITECNICO DI TORINO
Scuola di Dottorato
Doctorate in
Electronics and Communications
XXV cycle

Doctoral Thesis

DVFS using clock scheduling for Multicore Systems-on-Chip and Networks-on-Chip

A thesis submitted in fulfilment of the requirements
for the degree of Doctor of Philosophy

in the

Department of Electronics and Telecommunications

February 2014

Author:
Manoj Kumar Yadav

Supervisor:
Prof. Maurizio Zamboni

PhD Coordinater:
Prof. Ivo Montrosett

Abstract

A modern System-on-Chip (SoC) contains processor cores, application-specific processing elements, memory, peripherals, all connected with a high-bandwidth and low-latency Network-on-Chip (NoC). The downside of such very high level of integration and connectivity is the high power consumption. In CMOS technology this is made of a dynamic and a static component. To reduce the dynamic component, Dynamic voltage and Frequency Scaling (DVFS) has been adopted. Although DVFS is very effective chip-wide, the power optimization of complex SoCs calls for a finer grain application of DVFS. Ideally all the main components of an SoC should be provided with a DVFS controller.

An SoC with a DVFS controller per component with individual DC-DC converters and PLL/DLL circuits cannot scale in size to hundreds of components, which are in the research agenda. We present an alternative that will permit such scaling. It is possible to achieve results close to an optimum DVFS by hopping between few voltage levels and by an innovative application of clock-gating that we term as *clock scheduling*. We obtain an *effective* clock frequency by periodically killing some clock cycles of a master clock. We can apply voltage scaling for some of the periodic clock schedules which yield effective clock 1/2, 1/3, ... By dithering between few voltages we obtain results close to an ideal DVFS system in simple pipelined circuits and in a complex example, a NoC's switch.

Again in the context of a NoC, we show how clock scheduling and voltage scaling can be automatically determined by means of a proportional-integral loop controller that keeps track of the network load. We describe in detail its implementation and all the circuit-level issues that we found. For a single switch, result shows an advantage of up to 2X over simple frequency scaling without voltage scaling.

By providing each NoC's switch with our simple DVFS controller, power saving at network level can be significantly more than what a a global DVFS controller can get. In a realistic scenario represented by network traces generated by video applications (MPEG, PIP, MWD, VoPD), we obtain an average power saving of 33%.

To reduce static power, the Power-Gating (PG) technique is used and consists in switching-off power supply of unused blocks via pMOS headers or nMOS footers in series with such blocks. Even though research has been done in this field, the application of PG to NoCs has not been fully investigated. We show that it is possible to apply PG to the input buffers of a NoC switch. Their leakage power contributes about 40-50% of total NoC power, hence reducing such contribution is worthwhile. We partitioned buffers in banks and apply PG only to inactive banks. With our technique, it is possible to save about 40% in leakage power, without impact on performance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, non-traditional computing platforms, especially mobile and portable computing devices, have reached a level of computing capacity that equals the capacity of more traditional devices, like desktop computers. This evolution is driven by the need to support sophisticated and diverse applications, often running concurrently, aimed to improve the user's experience. At the basis of such evolution stands a supporting hardware infrastructure, the System-on-Chip (SoC). A modern SoC contains multiple general purpose processor cores and a number of application-specific processing elements, a fair amount of memory (typically SRAM), other than more or less standard peripherals, all integrated in a single chip. These elements need to be connected with a high-bandwidth interconnection system. Therefore standard busses, which do not scale in performance sufficiently well when the number of elements that are connected increases, are being replaced by packet-switched networks. A Network-on-Chip (NoC) offers a much higher bandwidth than a bus and also a significantly reduced latency.

The SoC concept is also moving from the domain of non-traditional computing platforms to the more traditional one, and it is now common to have specialized hardware blocks in a multicore CPU, connected to on-chip memory and regular processing cores with a high-performance communication infrastructure.

One of the problems that affect such large and complex SoCs is that all the interconnected blocks run each ideally at an optimal clock frequency, and often also at an optimal voltage, which are the values that optimize performance and power consumption. As a result, the SoC often is an asynchronous ensemble of components that run each in a synchronous way with their own local clock. This kind of system is called Globally-Asynchronous Locally-Synchronous (GALS) system. In the context of SoC with an NoC, the network is also the bridge between blocks running at different clock frequencies. An

NoC can be then designed as an asynchronous network [1] or, more frequently, as a synchronous network with its own clock [2].

The downside of the very high level of integration and connectivity in SoCs is the high power consumption. In mobile and portable systems, this problem is related to the duration of batteries, which is one of the key factors that affect the user's experience. In desktop systems, this is often related to reliability and thermal issues, an has mostly to do with the maximum power that can be dissipated while keeping the chip temperature below a given alarm threshold.

The power consumption in SoC implemented in CMOS technology is made of a dynamic component, which is proportional to the clock frequency and to the square of the supply voltage, and a static component, which is the product of the leakage current and the supply voltage.

To reduce the dynamic component, the method called Dynamic voltage and Frequency Scaling (DVFS) has been successfully applied to many commercial products, and consists in changing the clock frequency to track the system workload, and in changing the supply voltage together with the frequency in such a way to guarantee that the voltage is sufficiently high for a given frequency (i.e. such that no timing errors occur), but low enough to save power. DVFS is very effective because of the square relationship between voltage and dynamic power. Moreover, given that the way voltage is scaled is approximately linear with frequency and that dynamic power depends on the product of frequency and square voltage, DVFS makes the dynamic power vary almost in a cubic way with the clock frequency or, equivalently, with the supply voltage.

The static contribution to total power is also reduced with DVFS, but to further reduce it, the method called Power-Gating (PG) is used. PG consists in switching-off the power supply of blocks that are not in use via pMOS headers or nMOS footers that are connected in series with the block to be turned-off.

Although DVFS is very effective chip-wide, the power optimization of complex SoCs calls for a finer grain application of DVFS. Ideally all the main components of an SoC should be provided with a DVFS controller that sets the optimal voltage and frequency for that particular component. Fig. 1.1 illustrates this situation for an SoC in which blocks are connected via a NoC. It is clear from the figure that if each component runs at its own frequency, which is set by the local DVFS controller, the system is globally asynchronous and locally synchronous, i.e. a GALS system according to the previous definition. Re-synchronizing circuits are required to make data move from one clock domain to another. When voltage scaling comes into play, it adds a further layer of complexity and level shifters are required when data cross two voltage islands. Level

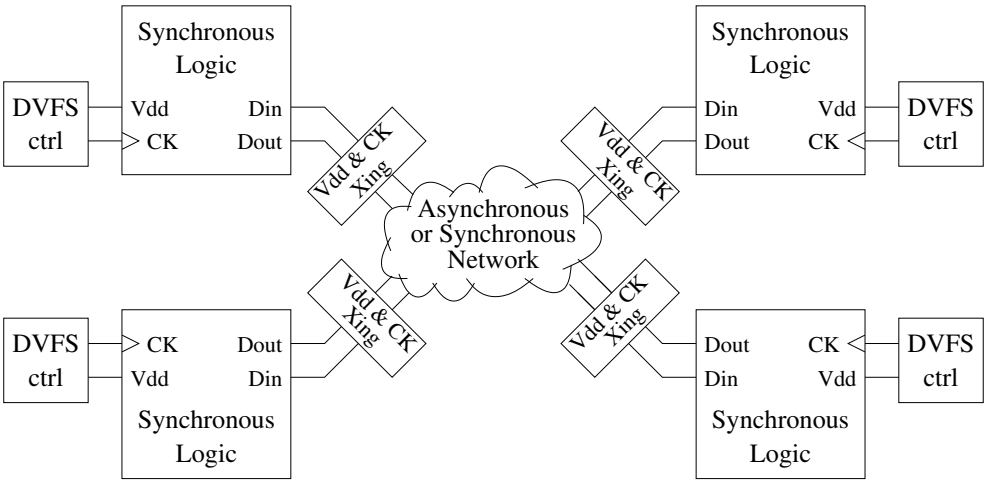

FIGURE 1.1: A GALS system with per block DVFS

shifters and re-synchronizing circuits are placed at the boundaries of the separate clock and voltage domains, which is where the blocks connect to the network ("Vdd & CK Xing" blocks in Fig. 1.1).

An example of this kind of GALS system with multiple voltage and frequency domains is the Intel 48-cores chip [2], in which processor cores and NoC are grouped into 8 voltage islands and 28 frequency islands. Two voltage reference controllers serve the 8 islands and react to voltage switch commands in about 1 ms. Frequency controllers react more promptly than voltage controllers do (about 20 clock cycles) and are more numerous because they are easier to integrate, although they consume a significant amount of power and area. This Intel's chip is a research-stage device, but it is foreseen that in a near future commercial products will contain hundreds of cores and specialized processing elements integrated in a single chip. What is less predictable is, however, how it will be possible to provide an individual, full-fledged DVFS controller for all those integrated elements.

The usual arrangement for DVFS with few voltage and frequency domains consists of off-chip or on-package voltage regulators paired with on-chip PLLs, as it happens in the mentioned Intel's 48-cores chip. To make DVFS scalable to hundreds of elements, voltage regulators need to be integrated on-chip [3]. Voltage and frequency controllers, however, consume power and area, because of the need to integrate passive elements like inductors and capacitors used in DC-DC converters [4]. Such passive elements, contrary to logic circuits, do not scale as technology progresses.

Based on this reasoning, it is then unlikely that each element of a core-rich GALS system will be fed by a complex and bulky DVFS controller. Designers of future systems are then left with two alternatives: grouping together many cores under the same controller, or seeking solutions that do not use standard voltage regulators. In this thesis we pursue

the second alternative and show that it is possible to design a simplified DVFS controller that gets rid of DC-DC converters and of PLL or DLL circuits. In particular, we observe that it is possible to achieve results close to an optimum DVFS by hopping between few voltage levels and by using a new frequency scaling method based on an innovative application of clock-gating. In Chapter 2, the method is explained in detail, whereas its application to pipelined circuits is described in Chapter 3. In Chapter 4 the application to the core element of an NoC, the switch, is presented, and in Chapter 5 the power saving obtained at the network level are presented discussed.

We have mentioned also power gating as an additional tool in the designer's toolbox for power reduction. Even though a lot of research has been done in the field of PG, the application of PG to NoCs has not been fully investigated. We tried to fill this gap by showing that it is possible to apply PG to one of the most crucial components of a NoC switch, the input buffers, hence reducing its static power contribution to the overall power, without a significant impact on performance. This contribution is described in Chapter 6.

Chapter 2

DVFS Based on Voltage Dithering and Clock Scheduling

This chapter focuses on the simplified DVFS scheme that we have devised. We first set the ground for our contribution by summarizing the key aspect of DVFS with few voltage and frequency levels, rather than ranges of infinite values between a lower and an upper bound. In particular we put the accent on the voltage dithering technique, which is shown to be effective and able to produce power saving close to what a regular voltage scaling technique can achieve. Then we focus on the contributions of our work:

- One of the key components of our simplified frequency scaling, is the *scheduler*. Instead of changing clock frequency, we schedule high frequency ticks of master clock through clock-gating. We obtain *effective* clock frequency by selectively and periodically killing some clock cycles.
- We discuss how voltage scaling can be performed in conjunction with clock scheduling, in such a way that timing errors are prevented. We can apply voltage scaling for some of the periodic clock schedules which yield effective clock 1/2, 1/3, ... We show that by dithering between few voltages, and using corresponding effective clock frequencies, we obtain power versus frequency behaviour close to that of an ideal DVFS system.
- We propose a further alternative clock-scheduling method based on distributed clock gating, to limit drawbacks of global clock-gating like current in-rush and consequent ringing of supply networks. This applies to pipelined circuits and uses stall signals that travels in direction opposite of data flow in the pipeline.

2.1 DVFS

Dynamic Voltage and Frequency Scaling (DVFS) is a technique by which supply voltage and operating frequency of a circuit under consideration are changed dynamically depending upon requirements of performance and energy saving. Ideal DVFS consists of infinite voltage and frequency pairs to choose from, one voltage for each frequency. This unique voltage level is the minimum voltage that makes circuit work without any error. The curve labeled "Ideal DVFS" in Fig. 2.1 made up of pairs of unique voltage and unique frequency shows behaviour of an ideal DVFS system, describing how dynamic power in a CMOS circuit scales when normalized frequency is varied from 0 to 1 and the optimal voltage is chosen for each frequency value.

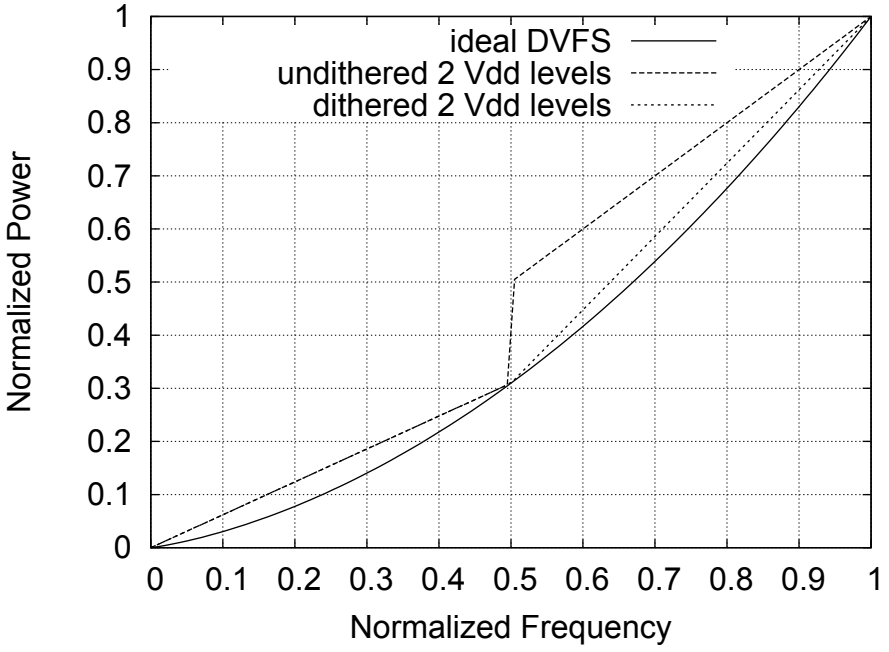

FIGURE 2.1: Power versus Frequency

However, a practical system has a finite number of voltage and frequency pairs to choose from. Also, frequency is not necessarily unique for each voltage available. The curve labeled "undithered" in Fig. 2.1, shows a case in which two voltage supply levels are used, and in correspondence with each of those two values, either the frequency range from 0 to 0.5 or the range from 0.5 to 1 is considered. In each range, the frequency is scaled but the voltage is kept constant. As a result, the dynamic power scales linearly in each range, proportionally to the frequency, with a slope that is set by the square of the voltage value in that range.

The DVFS method associated with the "undithered" curve in Fig. 2.1 is not very effective in comparison to an ideal DVFS method, especially in the upper frequency range. The

curve labeled "dithered", instead, is close to the ideal curve. It is obtained by a different DVFS method that consists in switching back and forth continuously, *dithering* that is, between two operating points. Each operating point is a pair of unique voltage and unique frequency. Thus there is a simultaneous change in voltage and frequency. The other points in the curve, different than the two dithered points, are obtained as a result of an averaging effect which produces an *effective throughput,* which depends on how long the system stays in each of the two operating points while dithering, and produces a correspondent average power. The straight line joining points (0.5,0.3) and (1,1) in figure 2.1 shows achievable average throughput and power by dithering normalized frequency between 0.5 and 1 (and the voltage is dithered correspondingly). One may also choose to dither frequency between 0 and 0.5, and the straight line joining points (0,0) and (0.5,0.3) is the average throughput/power line.

This method has been described in details in [5], [6], and [7], in which the authors show that by dithering among a few discrete voltage/frequency pairs, it is possible to get average throughput and average power close to what an ideal DVFS system yields, and that adding more voltage levels results in an increasingly better behavior.

For the dithering method to be effective, switching delay between voltages must be small, on the order of few clock cycles. Reportedly, in chip-wide implementations the transition between two supply voltages takes hundreds of microseconds [6] [7]. Calhoun and Chandrakasan show that "local" (instead of global) dithering using pMOS headers connected to two supply voltages needs one or few clock cycles [5]. They also report a small energy overhead. We therefore assume that switching overhead is small and tolerable if a single DVFS controller drives one block of a GALS system and not the whole chip. The dithering approach with two voltages and two pMOS transistors proved to be effective also in two more recent works [8][1]. Another advantage of such approach is that the circuit can be easily put into a low-leakage state by gating both power transistors (at cost of losing circuit's state).

In this Chapter, we will consider the case in which a limited number of voltages is used. Moreover, either frequency scaling or dithering between fixed frequency values is used when the required throughput does not correspond to the frequency value that corresponds to one of the voltages. To obtain the fixed frequency values a set of discrete values is obtained by scheduling via clock-gating the clock ticks of a master clock.

2.2 Throughput Scaling using Clock Gating

If we assume that a synchronous circuit executes a valid computation any time it receives a clock tick, its throughput is 1, that is one computation per clock cycle. Whenever clock gating is applied, some of the periodic clock ticks do not reach the circuit's registers that are forced to keep their previous state. The corresponding *average* throughput will be less than 1, less than one computation per clock cycle, on average.

Suppose now that a periodic clock-gating schedule is applied to the circuit in question. In a time period of $m+n$ clock ticks, m is the number of cycles passed through the gate, and n is the number of cycles stopped at gate. Then throughput of circuit is given by

$$\text{Thr} = \frac{m}{m+n} \tag{2.1}$$

Where m,n is 0,1,2,3,.... We include the case in which m is equal to zero, as this is what we define as "stop mode" case, which is useful when we want to stand-by the circuit. We also include the case in which n is equal to zero (but in that case $m > 0$) as this is what we define as "free running mode", because the circuit runs unstopped, and throughput reaches its maximum value of one operation per clock cycle.

Other than the corner cases in which m and n are zero, there are other important cases to consider. The most relevant for the discussion that follows is when m takes value 1:

$$\text{Thr} = \frac{1}{1+n} \tag{2.2}$$

Where n is 0,1,2,3,... In particular, if $n > 0$, (2.2) indicates that we have only one cycle passed through gate out of two or more cycles, and the throughput takes values such as 1/2, 1/3, 1/4, and so on. The distance between two passed clock ticks in terms of clock cycles is 2, 3, 4, and so on. This means that the circuits that elaborate between valid clock ticks have a longer time available for execution than in free running mode. Hence reduction in supply voltage may be considered.

To obtain throughput values greater than 1/2 and approaching 1, it is necessary to set $m > 1$. In practice, it is sufficient to set $n = 1$

$$\text{Thr} = \frac{m}{m+1} \tag{2.3}$$

and have m take values 2, 3, 4,..., so that the throughput follows the progression 2/3, 3/4, 4/5, and so on. In this case two or more consecutive clock ticks are passed through the gate. The circuit does not have more time to elaborate in a single clock cycle than in the free running mode. Therefore, circuit needs to be operated at nominal supply

voltage from throughput 2/3 onward. In our approach to DVFS, the 1/2 case (50%) is the highest throughput at which less than nominal supply voltage may be considered.

In standard frequency scaling, the throughput of a circuit is changed by tuning oscillator (clock generator) so as to obtain the desired frequency. In our technique, we vary throughput by applying clock gating to the stream of clock pulses produced by an oscillator. Thus, even though we let oscillator run at constant frequency all the time, we do vary throughput of the circuit under consideration. Therefore, we are creating an "effective clock" for the circuit, which consists of a scheduled block of $m + n$ pulses. The corresponding "effective clock frequency" is given by

$$\text{effective clock frequency} = \frac{m}{m+n} \text{ x } \text{ oscillator frequency} \tag{2.4}$$

In simple words, we may say:

$$\text{effective clock frequency} = \text{ throughput factor x } \text{ oscillator frequency} \tag{2.5}$$

2.3 Table of Schedules

An (m, n) schedule is, in our system, a pair of m and n values that produce an effective clock with a scheduled block of $m + n$ pulses. We selected sixteen different (m, n) schedule values that produce the throughput factors in Tab. 2.1 according to (2.1). The values of throughput ratio have been chosen with the intention to maintain a monotonic behaviour as well as reasonable distance between predecessor and successor values.

The sixteen schedules in table are indexed with a 4-bit binary code and have the following features:

- The Schedule table is almost symmetrical above and below schedule 0111 for number of cycles in a given schedule. Schedule 1100 is the only exception.
- Schedule 0000 corresponds to stop mode, and schedule 1111 to free run mode.
- Simplified ratio for schedule 1111 is 1/1. Similarly, for schedule 0000 it is 0/1. In table it is mentioned as 16/16 and 0/16 to show monotonic behaviour (increasing number of clock cycle above and below 50%) only. In fact, both schedules act on single clock cycles. Other schedules act on two or more clock cycles.
- Schedule 0000 has highest priority and can interrupt any schedule at any time before its completion. No other schedule can interrupt others.

TABLE 2.1: Schedules

schedule	pulse/out of	rate (%)
00 00	0/16	0.00
00 01	1/16	6.25
00 10	1/10	10.00
00 11	1/7	14.29
01 00	1/5	20.00
01 01	1/4	25.00
01 10	1/3	33.33
01 11	1/2	50.00
10 00	2/3	66.67
10 01	3/4	75.00
10 10	4/5	80.00
10 11	6/7	85.71
11 00	7/8	87.50
11 01	9/10	90.00
11 10	15/16	93.75
11 11	16/16	100.00

2.4 The Scheduler

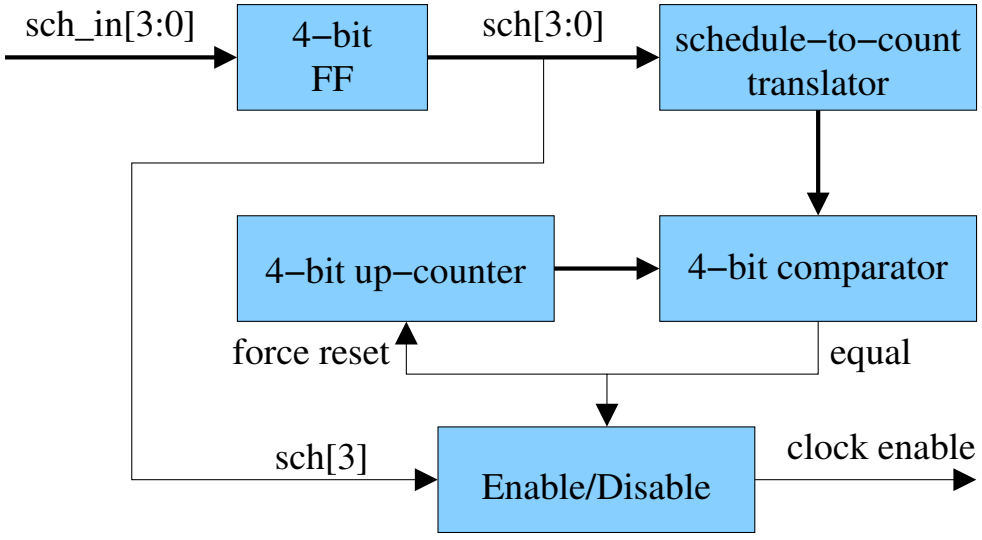

FIGURE 2.2: Block Diagram : Scheduler

To create a periodic clock gating signal in accordance with table 2.1, we devised a circuit named scheduler. As shown in block diagram in Fig. 2.2, the desired schedule is an input sampled by a 4-bit register on the clock's rising edge. The registered schedule is converted into a count value through a look-up table in block "schedule-to-count translator". The count value is made available to a 4-bit comparator that compares it with the value accumulated by a 4-bit up counter. If count value matches the counter's output, the comparator asserts the equality signal. This equality signal resets counter

on next clock cycle. Also, this signal is used to generate a clock-gating signal via the "Enable/Disable" block.

Count value is the number of clock cycles $m + n$ over which schedule is applied. This has been listed in table 2.1 as "out of", i.e. the denominator of the throughput fraction. For example, schedules 0001 and 1110 have count value of 16; whereas schedule 0111 has count value of 2.

Fourth bit of schedule determines whether clock cycles be enabled first and only last cycle be disabled, or it is the other way round: clock cycles be disabled first and only last cycle be enabled to pass through the gate. For short, "first enable and last disable" or "first disable and last enable".

If schedules higher than 0111 are applied, then scheduler will first enable all clock cycles to pass through the gate and will stop (disable) only the last one. For example, if schedule 1000 (fourth bit 1) is applied. Scheduler will apply clock gating signal over 3 clock cycles. It will let first two clock cycles pass through the gate (first enable), and will stop the last cycle at gate (last disable).

If schedule 0111 or lesser are applied, then scheduler will disable all clock cycles and will let only the last one to pass through the gate. For example, if schedule 0110 (fourth bit 0) is applied, the scheduler will apply clock gating signal over 3 clock cycles. It will first disable two clock cycles and will let the last one to pass through the gate.

Fig. 2.3 shows waveforms of the scheduled effective clock corresponding to all schedules.

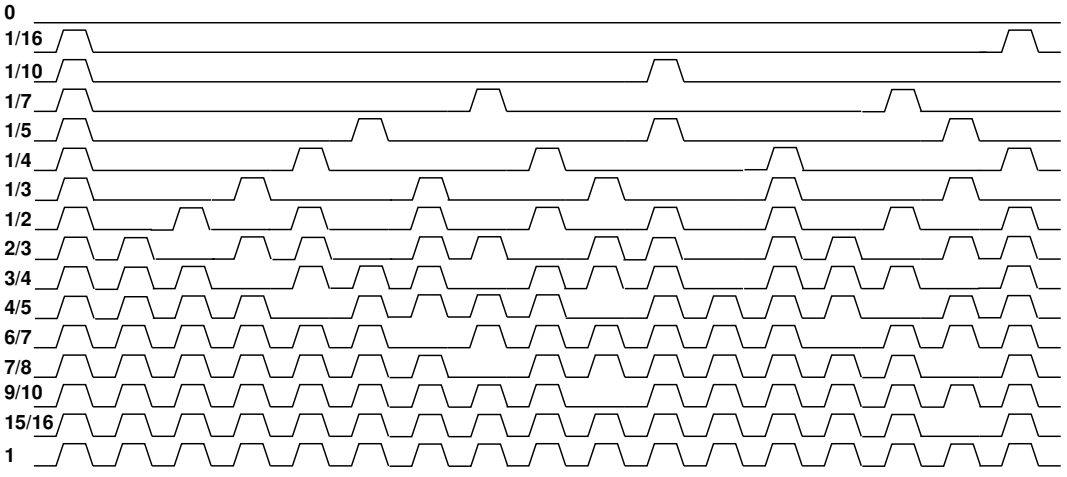

FIGURE 2.3: Sketch of all waveforms produced using clock gating

2.5 Standard vs clock gating based frequency scaling

Figure 2.4 compares standard frequency scaling with clock gating based frequency scaling. Nominal clock of period 1 unit has been compared with period of 2/3,1/2 and 1/3 times of nominal clock period. One may note that standard frequency scaling is achieved by proportional increase in ON and OFF times of clock. This, consequently, shifts rising and falling edges of clock relative to nominal clock's edges.

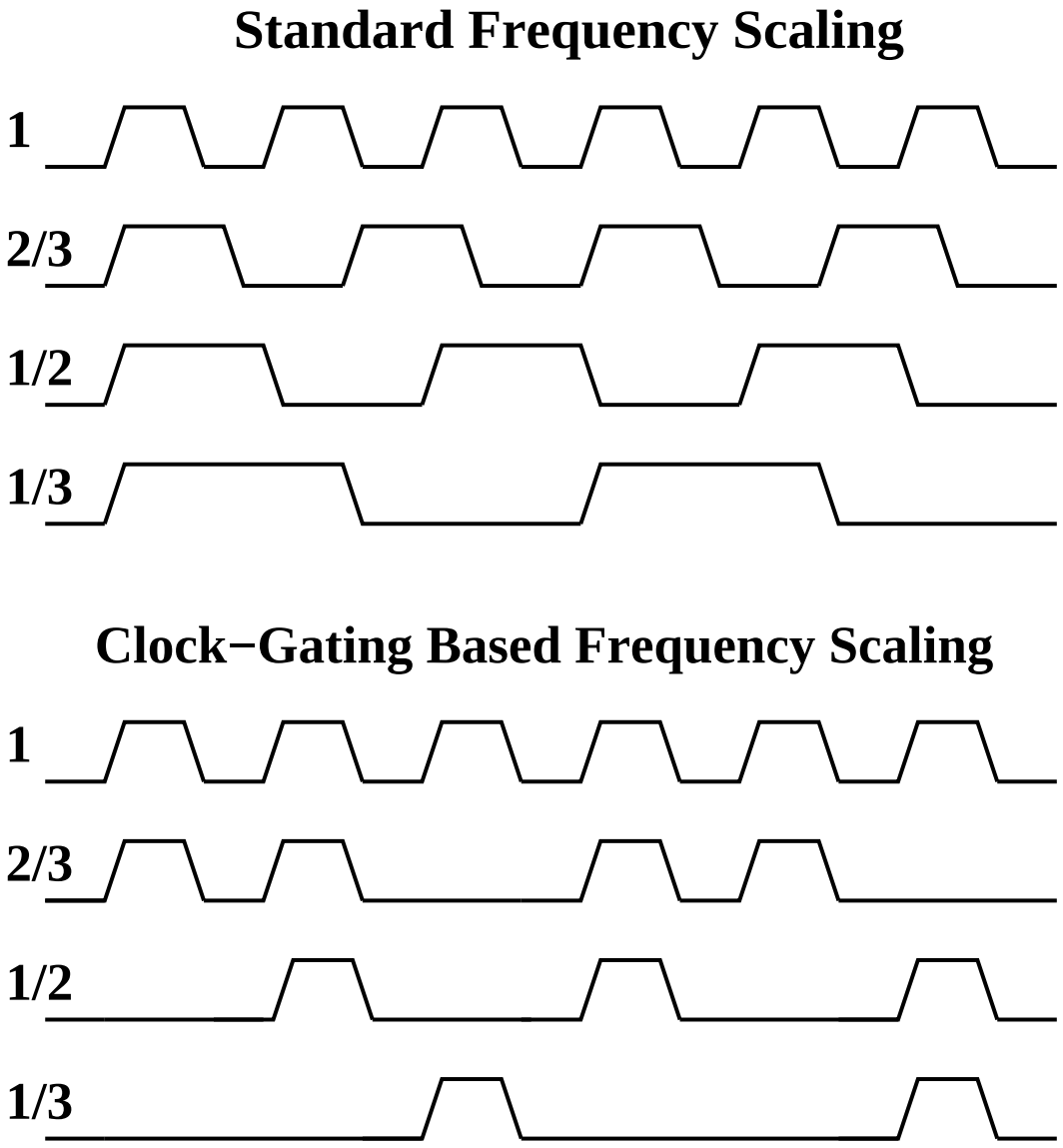

FIGURE 2.4: Standard vs Clock-gating based frequency scaling

In contrast with standard frequency scaling, clock-gating based frequency scaling is obtained by periodically killing (clock gating) ticks (pulses) of nominal clock stream (here marked as 1). For example, effective clock 2/3 has been obtained by allowing two pulses to pass through the gate and by killing (gating) third tick. Clock 1/2 has been obtained by gating first tick and allowing second tick to pass through the gate. Similar

analogy applies to clock 1/3 where first two ticks have been gated and third one has been allowed to pass through.

It is to be noted that there is no shift in position of rising and falling edges of effective clocks 2/3, 1/2 and 1/3 relative to nominal clock 1. Thus, circuit is always synchronized with oscillator in terms of phase. Thus in reference to a GALS system, where one master clock may serve all circuits, all circuits will remain synchronized with master clock in terms of phase. Interfacing two systems running at two different "effective clock frequencies" is then much simpler than interfacing two "real" different clock frequencies because we avoid any resynchronization issues[9][10].

2.6 Voltage Scaling

Figs. 2.3-2.4 make evident that the minimum distance between clock edges remain equal to one clock cycles for schedules of kind 2/3, 3/4, ... In these cases voltage scaling cannot be applied because propagation delays will grow and will create timing errors. This situation can be better explained in reference to a pipelined circuit like the one depicted in Fig. 2.5. We have a combinational circuit (CL) sandwiched between two pipeline registers. Data is provided (launched) by input register (REG in) to this combinational circuit to get it processed. Output of combinational circuit is received (captured) at output register (out REG).

A voltage scaling module having various level of supply at its input is shown at top of pipeline. This module consists of pMOS power switches to scale voltage among many available supply levels.

Clock gating based frequency scaling module has been deployed to serve effective clock to this circuit. This has been shown at bottom of figure with having only input of clock (clk). This module is made up of scheduler and clock gating circuit.

Let suppose our circuit runs at 1 GHz at the highest voltage and there is no delay margin (circuit delay and clock overhead sum up to 1 ns). We have applied schedule 1111 (throughput factor 1). As shown in upper sketch of Fig. 2.6, our circuit operates satisfactorily. If we now reduce supply level, we see a timing error. This error continues till schedule 1000 (throughput factor 2/3) as depicted in middle sketch of the figure. This is because, even through throughput has been reduced, time distance between two nearest clock ticks is still equal to that of nominal clock. Thus we need to increase time distance between launch and capture clock edges of input and output registers. In standard DVFS, clock frequency is reduced for this purpose. In our case, we must apply schedules 0111 or smaller (throughput factor equal or less than 1/2). We have least

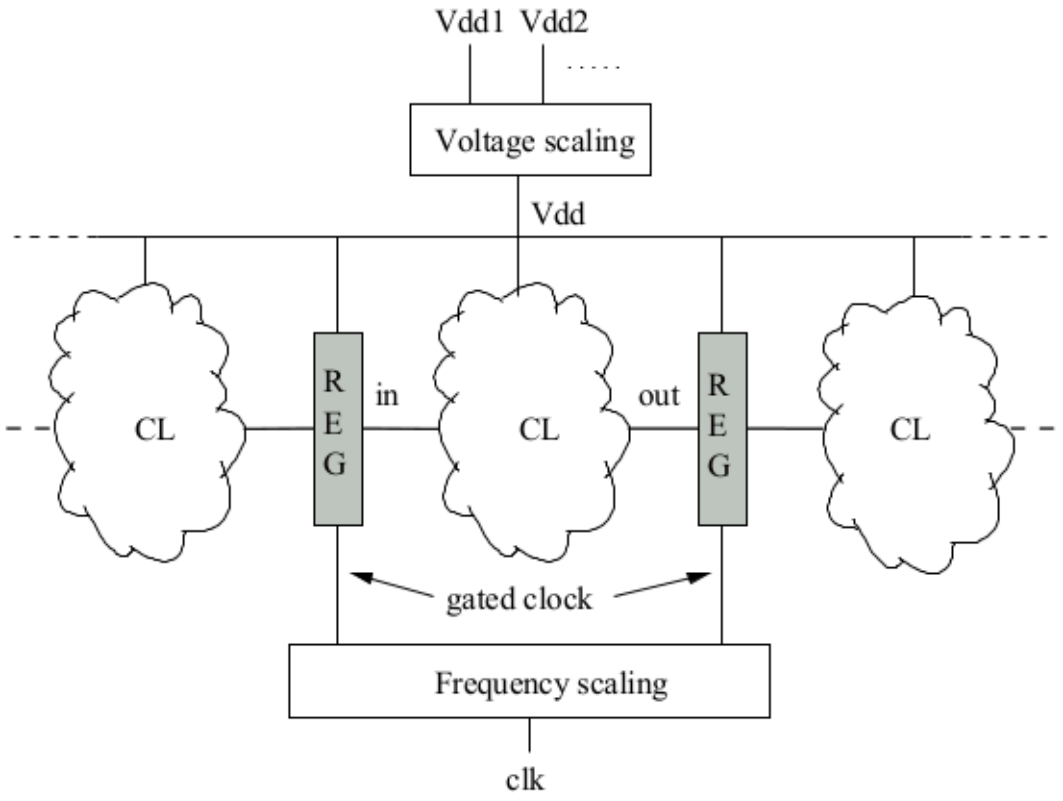

FIGURE 2.5: Diagram of pipeline with DVFS control
CL stands for combinational logic, REG is for register.

distance between two clock pulses equal to two or more times of period of nominal clock for these schedule. Hence, we may safely apply a lower supply voltage as shown at the bottom of the figure.

Scaled Vdd value can be obtained from curves like the one in figure 2.7. Here, the voltage that makes the delay twice or three times the one at the highest voltage (2 ns and 3 ns in figure) can be used for schedules 0111 (1/2) and 0110 (1/3). For the cases shown in figure, fastest pair is Vdd 1 and schedule 1111 (delay: 1 ns), followed by Vdd 2 and schedule 0111 (delay: 2 ns) and by Vdd 3 and schedule 0110 (delay: 3 ns).

Coordination between voltage and effective clock frequency change is crucial to avoid timing errors. Any time a voltage change is required, the scheduler clocks the gate (by temporarily setting the schedule to 0000) for the minimum number of cycles needed to have the voltage settle to the new value. According to [5] one or few cycles are sufficient.

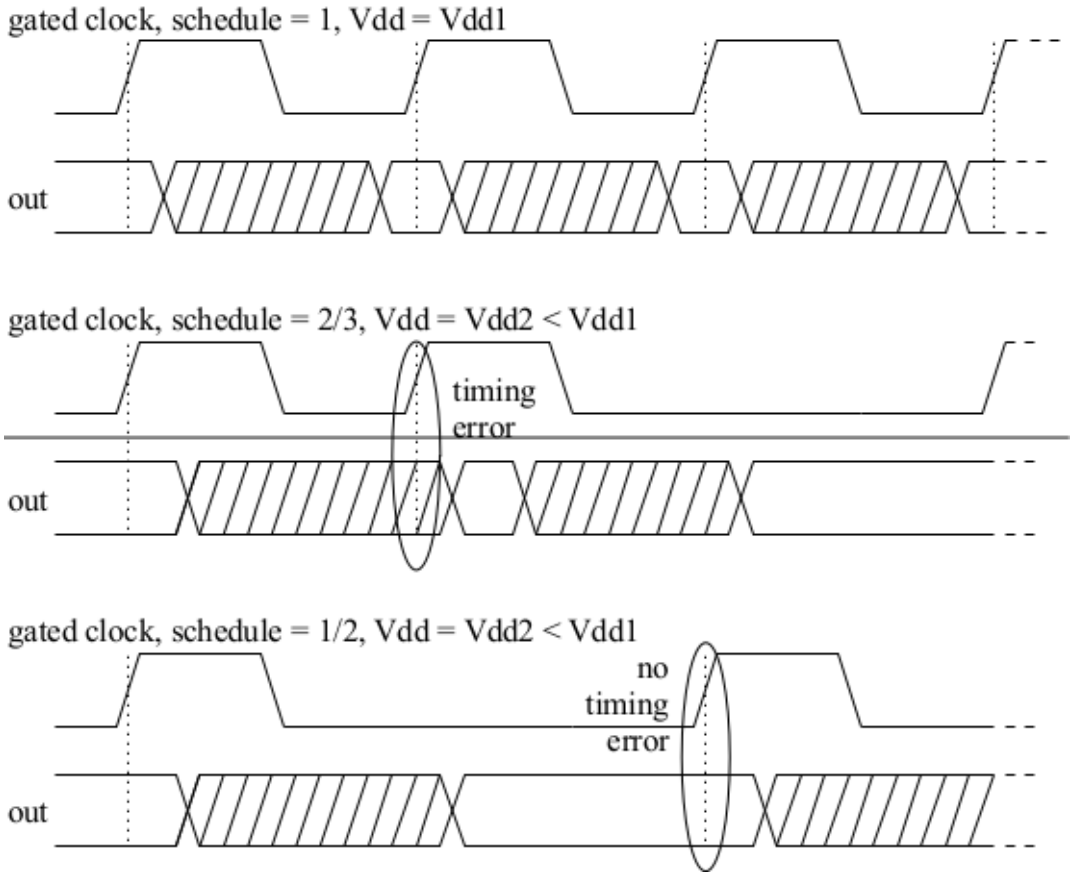

FIGURE 2.6: Supply level reduction and timing error

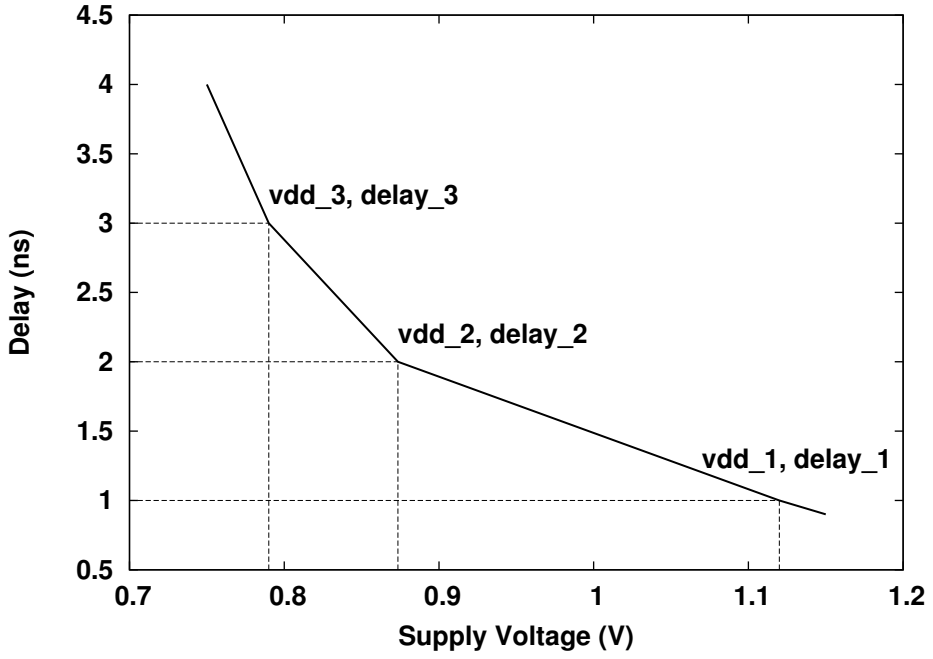

FIGURE 2.7: Typical Voltage vs Delay curve for a circuit in 45nm CMOS technology

Chapter 3

DVFS and pipelined circuits

We have seen in the previous chapter that our method for frequency scaling consists in using a scheduler that produces periodic schedules of clock ticks via clock-gating. For some of these schedules, it is possible to apply voltage scaling. In this Chapter we first analyze two possible methods for the detailed implementation of clock gating in pipelined circuits. Then we validate our contributions in reference to simple pipelined circuits and a more complex pipelined switch to be used in an NoC. We derive power figures and show that power saving obtained with our method in conjunction with dithering is close to what an ideal DVFS can get.

3.1 Clock Gating Mechanisms

We have considered two mechanisms of clock gating:

- Global Clock Gating
- Distributed Clock Gating

3.1.1 Global Clock Gating

A global clock-gating mechanism applied to a pipelined circuit, like the one depicted in figure 3.1, consists of a control circuit which controls all registers of the pipeline together. This controller uses scheduler (SCH in figure) and clock-gating logic (represented by the AND gate in figure) to stall all registers at once, or enable all of them together.

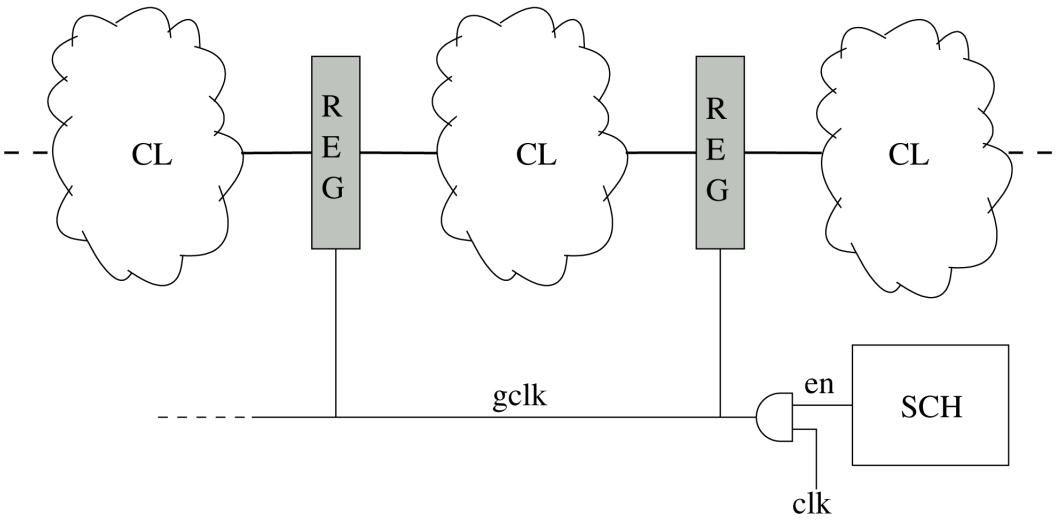

FIGURE 3.1: Global Clock Gating

3.1.2 Distributed Clock Gating

Application of global clock-gating may end up in problems like inductive noise and ringing of power supply lines because of current surge caused by fast turn-on/turn-off [11]. To prevent problems associated with sudden current inrush when global clock-gating is applied, a distributed clock-gating mechanism can be used. Every register in a pipeline stage is provided with its own small clock-gating controller. Scheduler sends enable signal to the last stage of pipeline. Controller of that stage enables the clock. It also forwards the enable signal to preceding stage on next clock cycle. In turn, the latter stage does the same with the stage that comes before it. In essence, the enable signal gets backpropagated and "pipelined".

To avoid loss of data when the enable signal goes low at some point in time and in a given pipe stage, one has to have two memory elements instead of a single register. First one is needed because as soon as enable signal goes down (stop), stopped stage has to have its output data on hold until withdrawal of stop. At the same time, one has to store newly arrived data. This new data is stored in second memory element and made available after the data on hold flows down in the system.

Distributed clock-gating mechanism can be implemented in two ways: using registers and using latches.

3.1.2.1 Register based distributed clock gating

In this mechanism, two memory elements required per stage consist of two registers and one multiplexer. This arrangement is shown in figure 3.2. When a disable signal is

applied, data of main register remains intact; while queue register accepts newly arrived data. Once data of main register flows down in the pipeline; stop signal is withdrawn. As soon as stop condition is withdrawn, content of queue register is made available to the data path to have it flow down.

This circuit, made of two memory elements along with control logic and a mux, is called Relay Station. This was introduced in context of SoC design to tolerate wire delays in pipelined interconnect [12]. Here we modify clock-gating controller to handle just a counter-flowing enable/disable signal; whereas in [12] there is also a validity signal flowing in the same direction of data which is immaterial for our purposes.

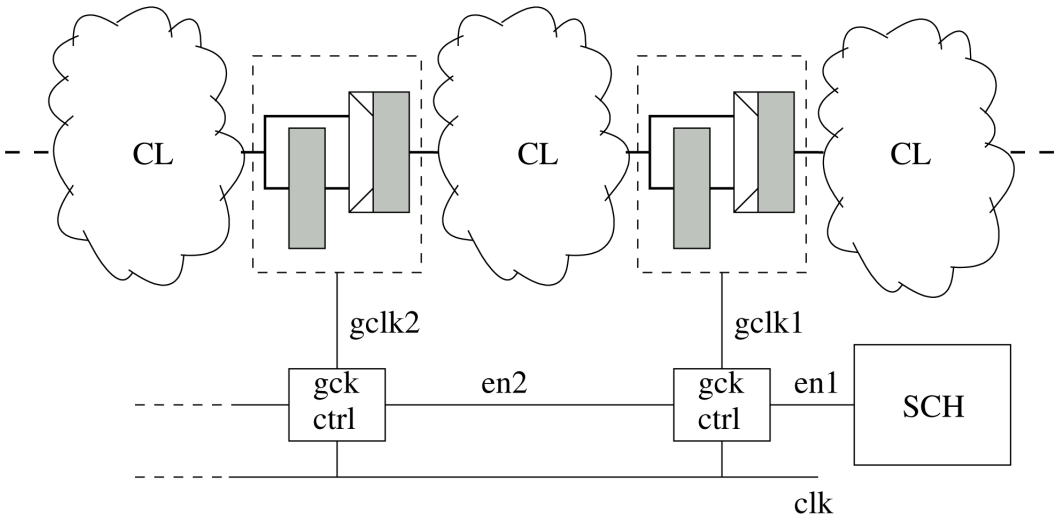

FIGURE 3.2: Register based distributed clock gating

3.1.2.2 Latch based distributed clock gating

This mechanism makes separate use of two latches which makes-up master-slave flip-flops. In this mechanism, data passes through two latches which also act as their place holder. When a stop condition is applied, latch acting on High-phase of clock, keeps on hold the data it already had. Latch acting on Low-phase of clock, accepts new data from preceding stage. Once data of High-phase flows down, stop is withdrawn. Thus, content of Low-phase latch replaces the content of High-phase one. Figure 3.3 shows this arrangement.

This smart method of using two latches was suggested first in [13] in the context of micropipelined circuits and then ameliorated in [14]. Exactly like in register based clock gating, we modified the clock-gating controller of the two latches to remove the validity signal.

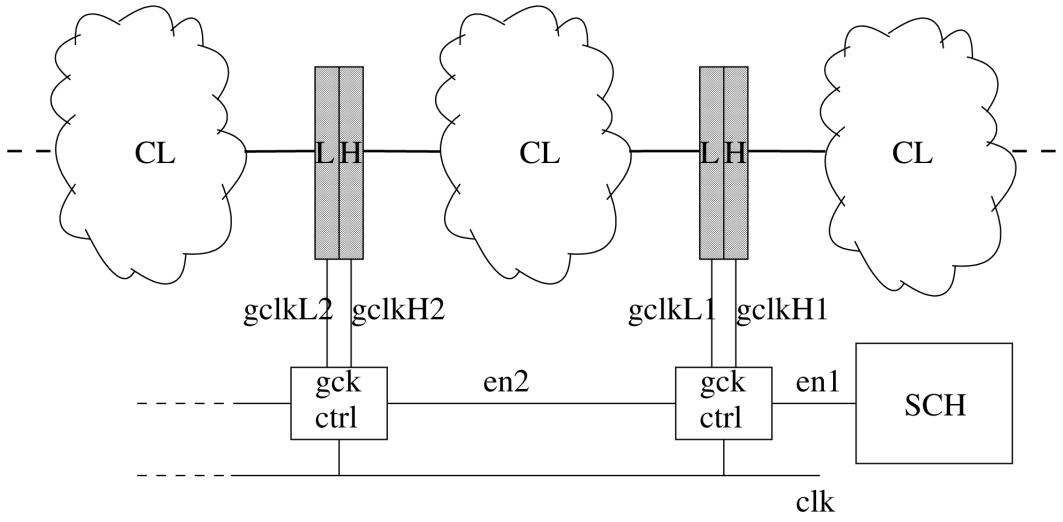

FIGURE 3.3: Latch based distributed clock gating

Two latches in a standard-cell library are usually bigger than one flip-flop. However, it is possible to design a new cell that is basically a master-slave flip-flop with separate access to its internal latches, with the only overhead of the space for the contact to access the extra pin for the secondary signal. It is worth noting that the two latches use a single-phase clock, so routing an additional clock tree is unnecessary.

3.1.3 Advantages and Disadvantages

Global clock gating requires only one control circuit to drive all stages of pipeline. Under distributed clock gating arrangement, each stage has its own small control circuit. This results in small overhead [14], however. This is the price to pay for smoothing power envelop when system passes from full clocking to no clocking. Full clocking to no clocking is the worst case for power supply to ring.

It is worthy to note that at any time, there is only one stage to be enabled or disabled in distributed clock-gating. This is advantageous because, large clock fanout in global clock-gating gets divided into many smaller fanouts of distributed clock-gating. This minimizes delay from input schedule to clock-enable, and simplifies interface of the logic under control of scheduler with other logic circuits clocked at the same frequency, but which do not use clock-gating.

3.2 Circuit Characterization at Different Voltages

As it was explained before, voltage scaling can be applied when effective clock frequency is 1/2, 1/3, . . . , of the full speed value. It is therefore necessary to evaluate the minimum voltage that can be applied at such scaled frequencies, given a target technology for our designs. All our experiments reported in this thesis are based on a 45nm CMOS technology from ST Microelectronics. We had libraries of standard cells that were characterized in terms of delay and power consumption at supply voltages of 1.15V, 1.10V, 1.00V and 0.95V. We had to extrapolate other voltages for our purposes. One of the extrapolated curves of delay vs voltage is shown in figure 3.4 for illustration.

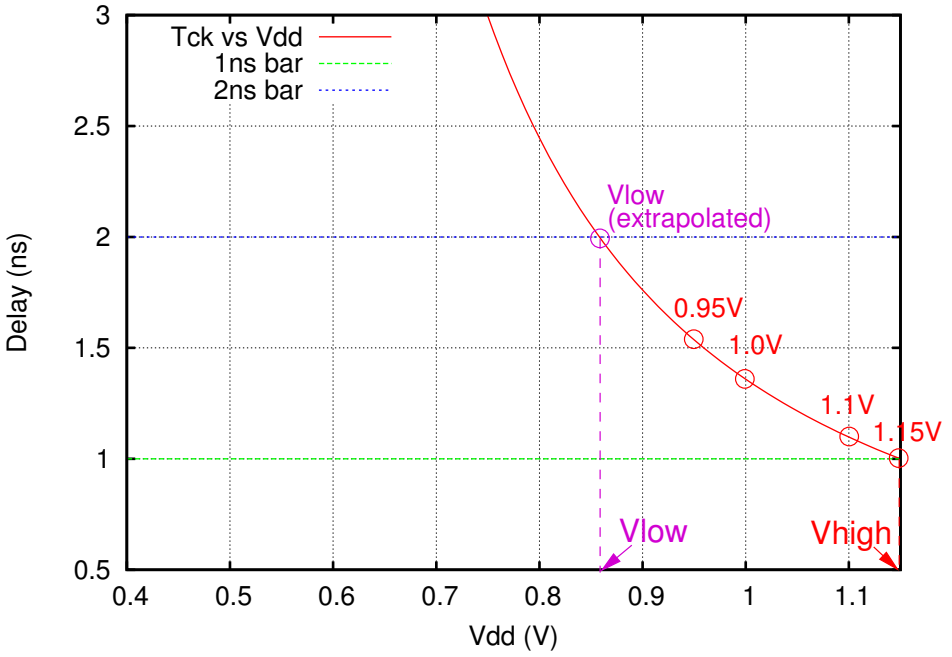

FIGURE 3.4: Extrapolation of Voltage

We obtained the curve by compiling and synthesizing first a sample circuit using Synopsys Design Compiler at highest supply voltage of 1.15V. We got Synopsys produce timing report using all four voltage libraries. With help of timing report, we calculated delay of critical path for each available supply voltage (1.15V, 1.10V, 1.00V, 0.95V). Let say, we had delay of Thigh at supply level of 1.15V. We plot all four delay points. We extrapolated supply level required for delay two times of Thigh (Tlow in figure), needed to run the circuit at schedule 1/2. The curve shows that 0.87 V (Vlow) can be used in this case. We did a similar extrapolation for delay three times of Thigh (not shown in figure) for the schedule 1/3, and determined that 0.73 V can be used for that schedule.

3.3 Validation of Proposal

To validate our approach we experimented first with simple pipelined circuits that perform 32-bits arithmetic additions; and then with a pipelined switch used in a NoC. For each circuit under test and clock-gating method outlined in section 3.1, our methodology consists of the following steps:

- Synthesis is performed targeting a standard-cell library characterized at the highest voltage and with clock constraint corresponding to the highest frequency.
- Timing analysis is performed with previously obtained gate-level netlist at different voltages. The critical path and its delay are recorded so as to build a curve of worst delay versus voltage (like the one in figure 2.7).
- For the ideal DVFS case, power is evaluated at every voltage/frequency pair obtained from previous analysis.
- In the delay/voltage curve, voltages corresponding to a clock period twice and three times the period at the maximum voltage are identified. (e.g. in figure 2.7, 1.1V for 1 ns is the highest voltage/frequency pair, 0.89V for 2 ns and 0.8V for 3 ns.) These voltages can be supplied to clock-gated circuits when effective clock frequency is 1/2 and 1/3 of the highest one (0.50 GHz and 0.333 GHz in the example in figure 2.7).
- Power consumption is evaluated varying the input schedule in the dithered and non-dithered cases. Power evaluation consisted in running a 1-ms simulation at gate-level (post-synthesis) with random input patterns and back-annotation of net activities into Synopsys Design Compiler.

In sections 3.4 and 3.5, we discuss results obtained for simple pipeline and switch, respectively.

3.4 A Simple Pipelined Circuit

Curves labeled Ideal DVFS in figures 3.5 and 3.6 represent optimal power obtained when an ideal DVFS controller feeds our simple, three-stage pipelined adder. In Fig. 3.5 global clock gating was used. In Fig. 3.6 distributed clock gating was used in the version with latches.

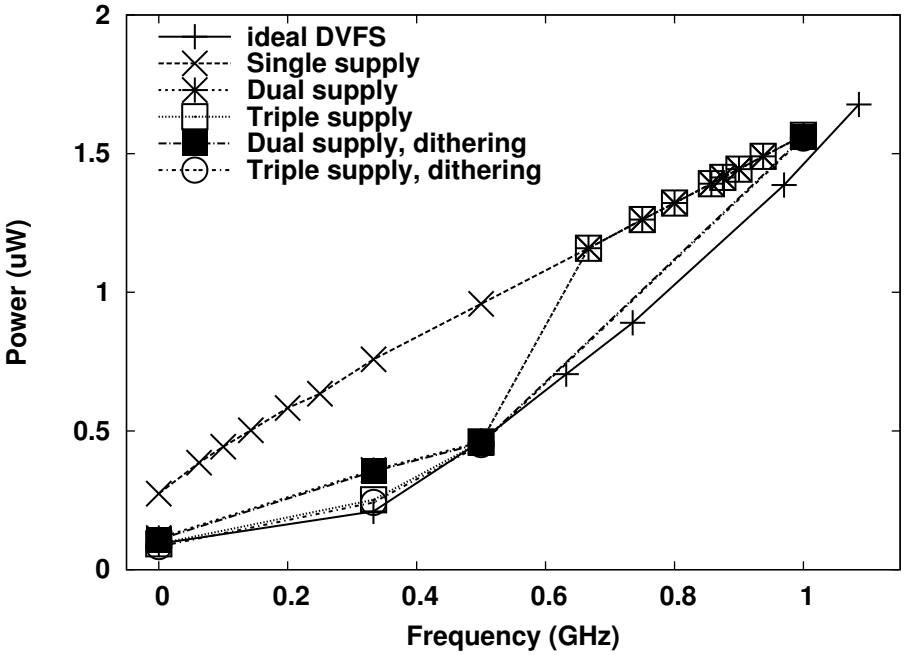

FIGURE 3.5: Pipelined Adder with global clock gating

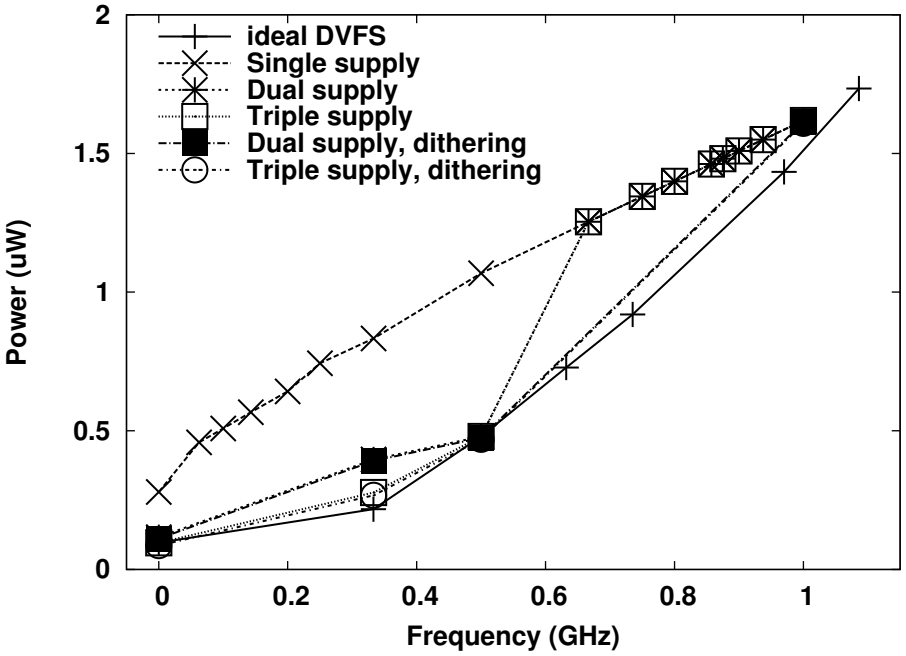

FIGURE 3.6: Pipelined Adder with latch based distributed clock gating

In that case x-axis represents real clock frequency and not the effective clock. Other curves are for power versus effective clock frequency. For each type of clock-gating, five cases have been considered: one, two and three supply voltages, and, with and without dithering.

Single supply curve is obtained by keeping voltage at 1.15V and changing schedule from 16/16 down to 0/16. Refer table 2.1 for list of schedules. As a result, power decreases almost linearly. This is the expected behaviour since dynamic power is proportional to clock frequency. However, power scaling behavior is largely suboptimal compared to ideal DVFS.

Dual supply curve without dithering is obtained with 1.15V in schedule range 1 down to 2/3 and hence it superimposes with single supply curve in this range. In range 1/2 down to 0 we can safely reduce supply voltage, and so 0.87V was used.

Triple supply curve without dithering is obtained like dual supply curve in range 1 down to 1/2 (hence superposition of two curves), and with third voltage equal to 0.73V in schedule range 1/3 down to 0. In lower range 1/2 down to 0, the two curves get close to the ideal one, especially triple supply curve. These are the best possible results without dithering.

If dithering between two or three voltages is possible, we obtain better results, as clear from Dual supply, dithering and Triple supply, dithering curves. In lower range there is no difference with corresponding curves without dithering, because they have been obtained in the same way. In upper range 1/2 to 1, dithering between 1.15V and 0.87V gets us close to the ideal DVFS curve. It is clear from graphs that even two supply levels are just sufficient for close to optimum power saving with dithering. Having a third supply level, one can also save power in lower frequency range.

As far as difference between global and distributed clock gating goes, power results are similar, with a larger power consumption of the distributed case between 3% and 10%. Obtained results are in agreement with expected behavior of figure 2.1.

In contrast with latch-based clock-gating, if we use register (also called relay station) based clock gating to implement distributed clock-gating, we would obtain a totally different behavior, as indicated in figure 3.7. Power consumption without dithering is not monotonic with frequency and exhibits a power peak.

To understand behavior of register based distributed clock gating, we have to look at its activity in response to enable/disable signal received from scheduler. For all schedules between 50% and 100%, with each withdrawal of disable signal, data move from queue register to main register. As a result, when we increase number of disable signals to

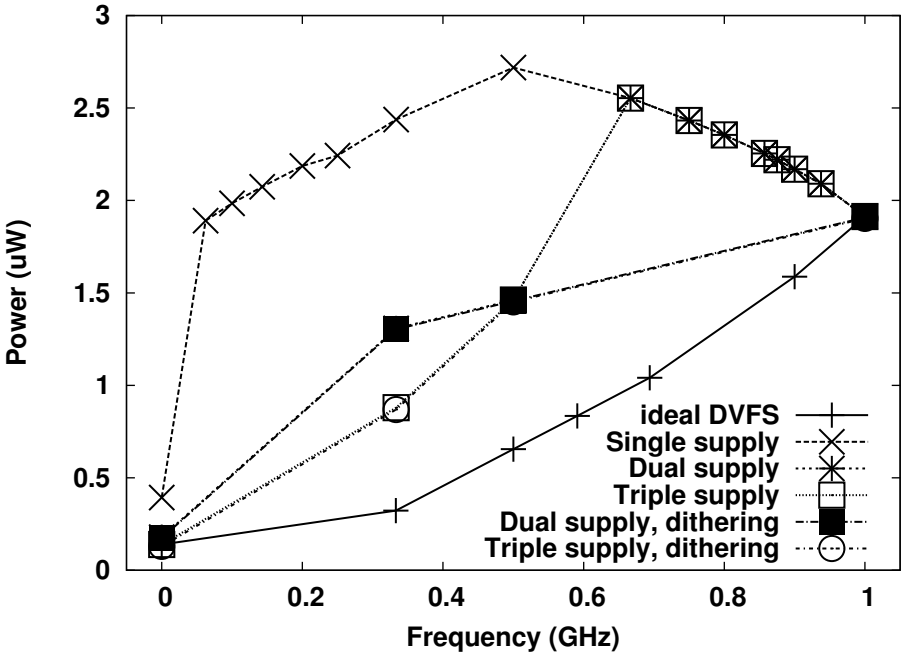

FIGURE 3.7: Pipelined Adder with register (or relay station) based distributed clock gating

reduce effective frequency, we are actually increasing internal activity; hence power increase. When we choose a schedule of duty less than 50%, we enable the pipeline only once after two or more clock cycles. Hence we are decreasing activity by prolonging withdrawal of disable signal. Thus we see a decline in power consumption. A sudden fall in power in dual and triple supply cases is due to change in supply voltage. Since schedules in range 2/3 to 1 are not used in the dithering cases, resulting curves do not show any peak. Nonetheless the power behavior is far from an ideal DVFS.

From this first set of experiments, we conclude that we can rule out distributed clock-gating mechanism based on register (or relay stations). Thus, for next set we only considered global clock-gating and latch-based distributed clock-gating.

3.5 A Pipelined NoC Switch

We evaluated power consumed by a pipelined switch that we designed for a NoC used as interconnect fabric for a GALS system. As it is shown in Fig. 3.8, our switch is connected to four other switches located on north, east, south and west. All the switches are arranged to form a mesh topology. Since the switch is also connected to a processing element or a memory block, it has in total five input and five output ports, each 64-bits wide. Flow control uses wormhole technique whereas routing uses the XY dimension ordered method. It has been proven that it avoids deadlock [15]. Elementary unit of

flow control is a 64-bit *flit* which can be sent or received in one clock cycle. Multiple flits (in variable number) form a packet. Figure 3.8 shows the switch's internal architecture which is derived from the architecture in [16] and consists of three pipeline stages. First stage is for link traversal and FIFO buffer write. Second one is for arbitration and routing pre-computation. And third one is for switch traversal (through a non-blocking crossbar). The pipelined nature of the switch makes it an ideal candidate for application of the methodologies described in the previous sections.

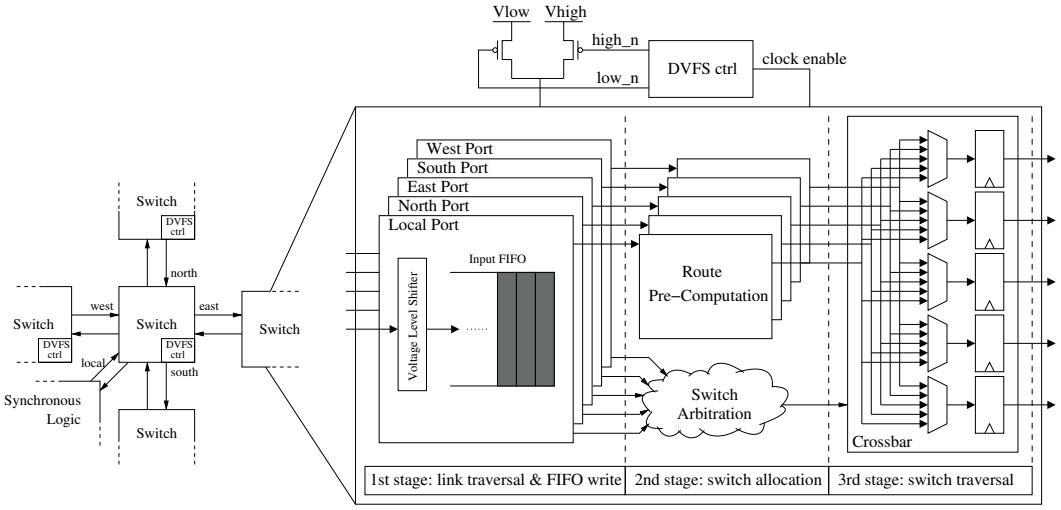

FIGURE 3.8: A NoC Switch

DVFS controller of switch schedules pulses of a master clock via clock-gating to obtain an effective clock frequency. It selects appropriate supply voltage through pMOS headers. It interfaces through 64-bits links with a local computational resource and with four other switches (located on north, west, east and south), each potentially running at different effective frequency and voltage. Resynchronizing FIFOs and voltage level shifters are used to permit correct interfacing. Figure 3.8 shows switch connections and reveals its internal structure that consists of three pipeline stages. First stage is for link traversal and FIFO buffer write. Second one is for arbitration and routing pre-computation. And third one is for switch traversal (through a non-blocking crossbar).

In global clock-gating implementation, scheduler disables clock simultaneously for all pipeline registers. In distributed case, scheduler applies its enable signal to third stage. Local clock-gating controller of third stage forwards enable signal to second stage after one clock cycle and in turn second one sends it to first stage, after two clock cycles. In both global and distributed cases, input FIFO write pointer has an autonomous local clock-gating circuit to avoid dropping incoming input packets when scheduler disables clock. When switch is clock gated, an output invalidation mechanism prevents surrounding switches and local resource from sampling its outputs (the most significant bit of output data is used as validity bit).

DVFS controller schematized in figure 3.8 selects among only two voltages, Vhigh and Vlow. However, we evaluated power consumption also for case of three voltages, with and without dithering, other than the single voltage and the ideal DVFS case. We tested both global and distributed clock-gating and for this last case, we only considered the latch-based implementation, having previously ruled out relay station method.

We implemented this switch in 45 nm technology used for previous set of experiments. Procedure outlined in previous section allowed us to estimate clock frequency of 0.923GHz at the highest voltage of 1.15V. It is then possible to scale voltage down to 0.89V for an effective clock frequency half the value at 1.15V (0.923/2=0.4615GHz), and to 0.75V for an effective frequency one third of the highest value (0.923/3=0.3077GHz).

Trends in figure 3.9 and 3.10 are in accordance with our expectations; and similar to those of simple pipelined adder case. Results highlight though a less effective scaling of power at a single supply voltage with respect to previous case: Ratio between power at full speed and at zero activity is 2.5x whereas it was 6x for simple pipelined adder. Especially in upper schedule range, from 2/3 to 1, scaling frequency is not really effective. Main reason is larger impact of leakage power in this more complex circuit. Two voltage and three voltage cases without dithering suffer from same problem in that upper range, because voltage used for those schedules is still 1.15V. Once again, dithering is the best option resulting in power curves that track well with the ideal DVFS case. Difference between global and latch based distributed clock-gating is small. Since there is no significant overhead for slightly more complex distributed approach, we recommend use of this method.

Cost of controller is small 0.9% in power and 0.4% in area of the switch. Therefore our approach to DVFS is very cost efficient.

3.6 Summary

We have discussed a simplified DVFS mechanism that uses voltage dithering among few levels (two or three voltages) for dynamic voltage scaling, combined with a frequency scaling method based on clock scheduling. This method is intended for applications in GALS systems using an aggressive, per block power management strategy.

To validate our technique, we have designed some circuits in a 45 nm CMOS technology and characterized them with standard synthesis and power evaluation tools. We have compared power versus frequency curves with those obtained by an ideal DVFS system (one that chooses voltage and frequency in a continuous range).

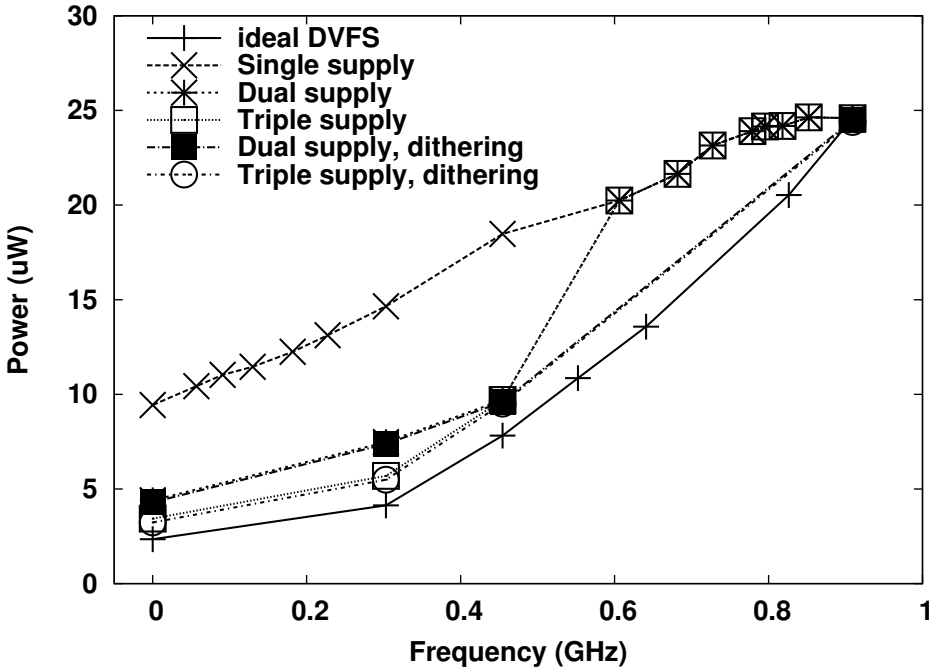

FIGURE 3.9: Switch with global clock-gating

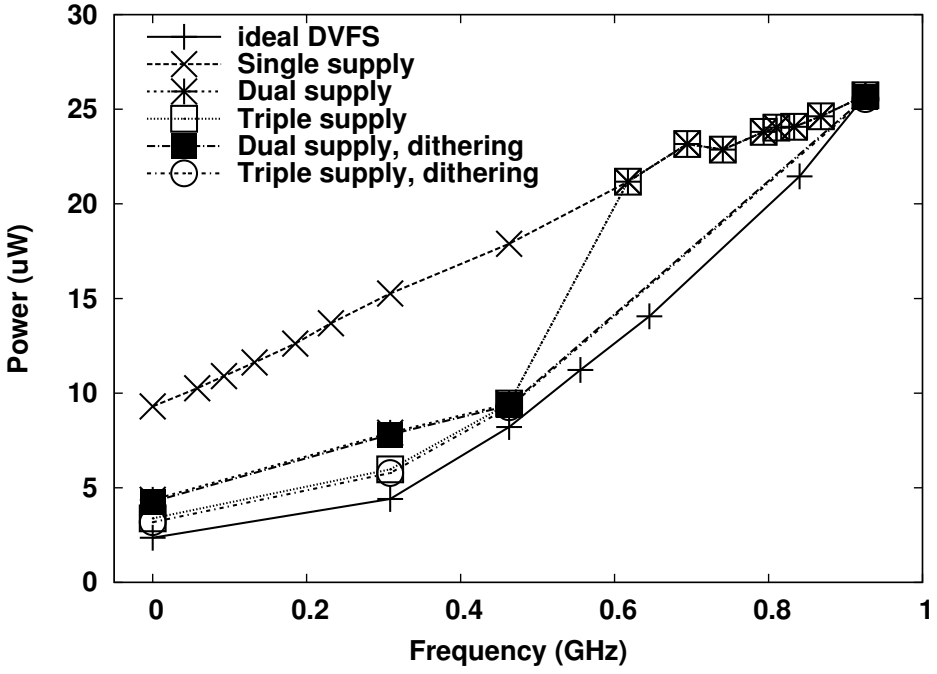

FIGURE 3.10: Switch with latch based distributed clock-gating

We have evaluated two different implementations of clock scheduling. First method couples clock scheduler with a global clock-gating circuit. Second one uses a distributed clock-gating approach. For this method, we have also evaluated two alternative implementations, register-based and latch-based. We have found that the register-based clock-gating method is not effective in reducing power. Latch based distributed clock-gating and global method produce similar results and are able to track well with power versus frequency curve of an ideal DVFS system.

Chapter 4

A Simple DVFS Controller for a NoC Switch

We have illustrated in the previous Chapters a simplified DVFS method that scales frequency using a simple circuit, which we called the scheduler, and associates with the scaled frequency a scaled voltage, which we vary in steps using two or three fixed values. We have not seen yet, however, how the schedule and the associated voltage can be chosen in such a way to minimize power while guaranteeing that the circuit workload is executed. In this Chapter we offer a solution to this problem in the practical context of an NoC. In particular, we focus on the switch element of a NoC to which we apply a simple DVFS controller based on the findings that we have previously discussed. In summary, these are the main contributions of our work that are illustrated in the following:

- We propose a simple DVFS Controller for a NoC Switch. This controller uses two voltage levels to scale voltage. Two pMOS transistors are used to connect the switch to high or low voltage supply, as suggested in [5] [1] [8]. Latency of switching between voltages is low, few clock cycles. Frequency is scaled by clock gating by means of our clock scheduler.
- Our DVFS controller leverages a control feedback loop to determine the most appropriate effective clock frequency such that the switch is able to sustain input traffic. We took inspiration from [17] which proposes a similar approach in context of a DVFS controller for a processor.
- We also solve problem of asynchronous communication between different switches in an NoC because all switches are effectively synchronized with a single reference clock.

4.1 Problem Statement

Network-on-Chips (NoCs) consume a relevant fraction of chip power [18]. To reduce it, dynamic voltage and frequency scaling (DVFS) can be globally applied to the whole network [2]. Voltage and frequency must guarantee the bandwidth requested by the network traffic [19]. Choosing voltage and frequency for the entire network requires, however, a global controller to measure local quantities, such as load of network nodes [20], an approach that does not scale well with the NoC size.

Besides, in traffic scenarios like the MPEG4 example in Figure 4.1, a local approach outperforms by far a global one [21]. Here, load is high only for three switches out of twelve. Global voltage and frequency are set by the highest load, even though many switches can run at a smaller frequency, and so at a lower voltage. The histogram in Figure 4.1(c) shows that Global and No-DVFS have the same power consumption.

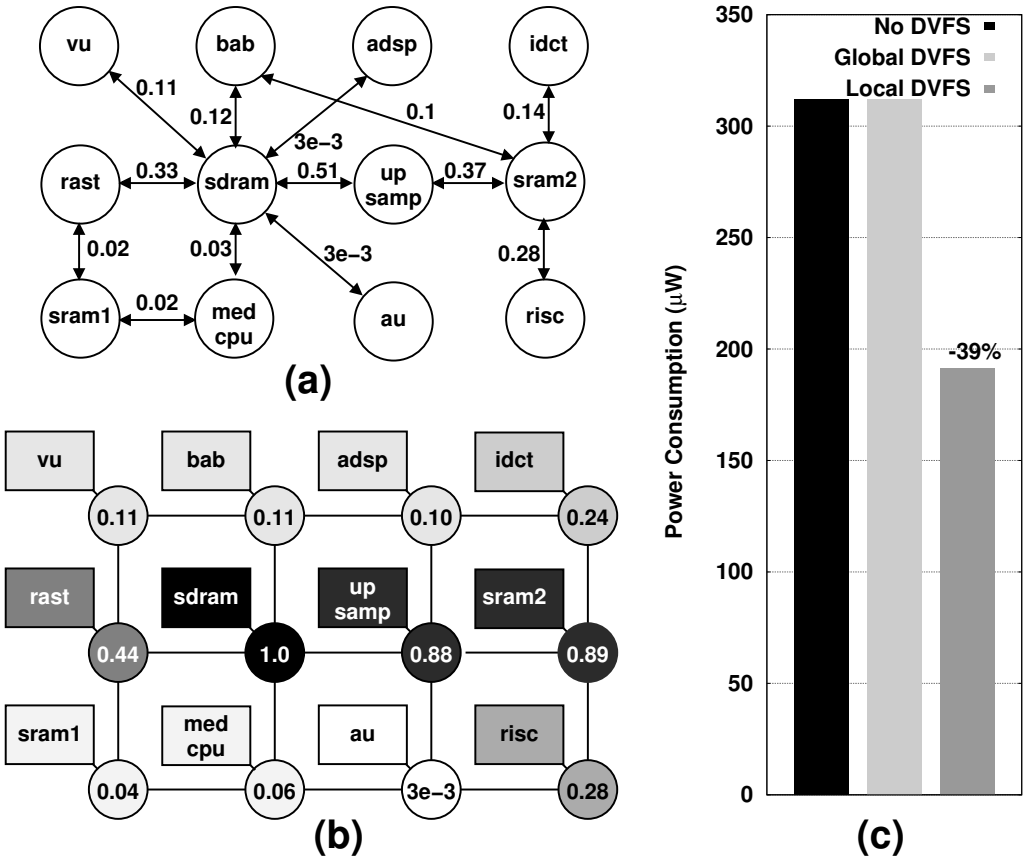

FIGURE 4.1: Local versus global DVFS in an NoC. (a) Communication graph in MPEG4 annotated with normalized rates. (b) MPEG4 tasks mapped to a 12-switches NoC, gray shading proportional to normalized switch rates. (c) Power consumption: The local approach DVFS tracks better the local traffic by controlling voltage and frequency of individual switches.

A local approach to DVFS, however, has practical limitations. First, even though integrated DC-DC converters and PLL/DLL for individual cores in multicore chips are investigated [3], it is unlikely that they be used in both cores and NoC switches [22]. Second, latency and power overheads for clock-domain crossing in the links between switches, each running at its own frequency, are significant [20] [23].

Our DVFS method solves these problems. If we go for a local implementation of DVFS, it is because of the simplicity of the scheduler–only a few gates are necessary to implement it–and the circuit for changing the supply voltage–based on pMOS transistors and few other gates, as shown in the following. This approach also solves the asynchronous communication issue, because all switches are synchronized with a single master clock, from which the effective clocks for each switch are derived.

4.2 Switch Architecture

We already succinctly presented the switch architecture at the end of the previous Chapter. For reading convenience we report one more time the architectural scheme in Fig. 4.2 and comment on some aspects that we have not touched before and that are important for the present discussion.

We assume that the DVFS controller located in each switch can autonomously choose between two voltage levels. Therefore, at the inputs of each switch are located voltage level shifters the permit correct interfacing with other switches that may be running at different voltages. The output of the voltage level shifters go into the input buffers, which consist of FIFO queues able to temporarily store up to 16 flits. When full, the FIFO sends a signal back to the sender (a switch or a processing or memory resource) to stop it and so to keep the data on hold which otherwise would be lost.

DVFS controller in figure 4.2 sets the voltage by issuing high_n and low_n commands (n stands for active low) and generates an "effective" clock frequency based on the clock-gating mechanism outlined in previous chapters.

4.3 Switch Effective Clock Frequency

The DVFS controller determines the effective frequency as the minimum one that makes the switch able to sustain the input traffic rate. The rate is estimated by a control loop feedback mechanism that attempts to minimize a suitable error function. Such error is the difference between the maximum occupancy and the desired occupancy of the input FIFOs. Figure 4.3 reports a block-scheme of the control system.

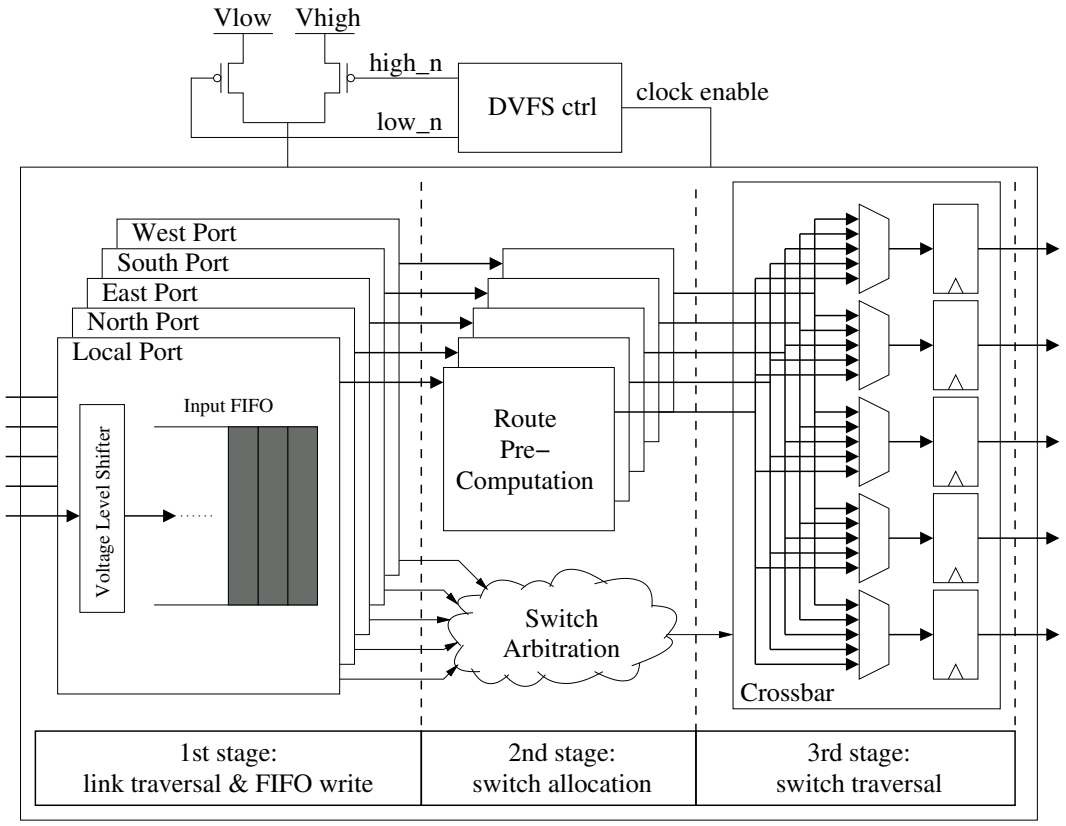

FIGURE 4.2: Architecture of the NoC switch with three pipeline stages and DVFS controller

The maximum among FIFOs' instantaneous occupancy is selected (MAX block) and averaged through a cumulative moving average (AVG). Such average helps filter out fast variations in occupancy so as not to impair the control loop. The simple cumulative moving average (CMA) method was selected because it requires the least amount of memory. In fact, for a 16-elements average, the CMA of the FIFO queue content at time n, $qCMA[n]$, is computed as follows [17]

$$qCMA[n] = \frac{15qCMA[n-1] + qMAX[n]}{16}$$

where qCMA[n − 1] is the previous value at time n − 1, and is the only term to be memorized, and qMAX[n] is the instantaneous maximum FIFO occupancy at time n.

The desired occupancy, qOBJ , which in our case is 8 flits, gets subtracted from the averaged maximum occupancy. The PI block in figure 4.3 implements the proportional-integral gain elements of a control loop, a variant of the classic proportional-integral-derivative PID controller. The output of the PI block, which feeds the scheduler (SCH), is an estimate of the maximum input arrival rate μ[n] (i.e. the maximum among the

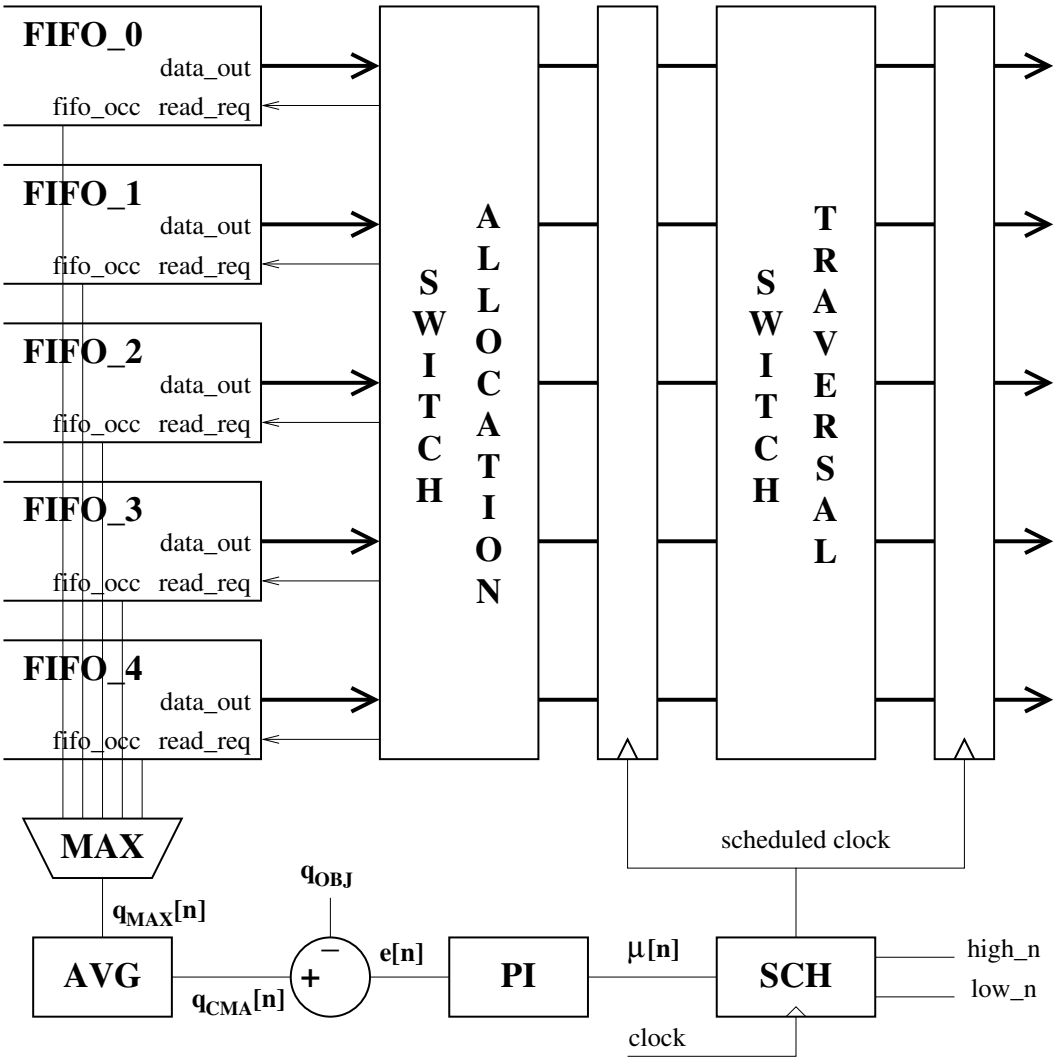

FIGURE 4.3: Feed back loop for automatic DVFS control

number of flits per clock cycle received by any input port) and is given by

$$\mu[n] = \mu[n-1] + K_I e[n] + K_P(e[n] - e[n-1])$$

Where e[n] = qCMA[n] − qOBJ is the error, K_P is the proportional gain and K_I is the integral gain of the controller. The two gains are chosen in such a way to guarantee stability of the loop [17]. The PI block contains two registers to store the old rate estimate μ[n - 1] and the old error e[n - 1]. The smaller the average target occupancy qOBJ, the smaller the latency of the switch at steady state. We experimented with different values, and found out that the control system works well even with 1 flit, which we finally set.

Computations of average, error and input rate estimate are carried out in fixed point fashion using twelve bits for the integer part and eight bits for the fractional part. The

scheduler block takes the integer part of the estimated rate μ[n] and applies a saturating function to it so as to reduce it to a 4-bit value. The scheduler contains a counter that increments at any clock cycle and gets reset when it reaches the desired periodicity of the effective clock frequency it has to synthesize. For instance, if the desired frequency is 1/3 or 2/3 of Fck, the counter runs for three cycles, from 0 to 2. A clock-gating mechanism is used to allow clock pulses to go through or to get stopped. In the previous example, for the 1/3 case the scheduler lets one pulse go through out of three, for the 2/3 case it lets two pulses out of three. In Table 2.1 we previously reported the 4-bit schedule code, the number of pulses and the periodicity for a given schedule and the corresponding rate as a percentage of the full speed.

The two-voltage levels are associated with the scaled clock frequencies in a simple way. The low voltage is used for all the schedules that generate a throughput of 1/2 or less. The high voltage is used in all the other cases. Fig. 4.4 makes this association evident.

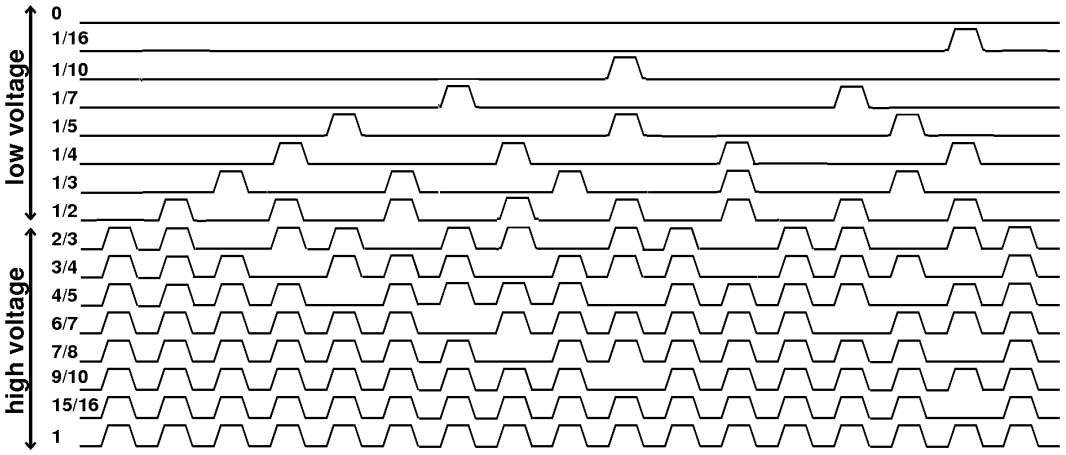

FIGURE 4.4: Sketch of all waveforms produced using clock gating and association with voltage level

The synthesized clock frequency is sent to the pipeline registers of the switch, as clear from Figure 4.3. The input FIFO write pointer has an autonomous local clock-gating circuit, not controlled by the feedback loop, which avoids dropping incoming packets when the scheduler disables the clock.

The switch was described in fully-synthesizable VHDL and was simulated under various conditions to check correctness as well as responsiveness of the control loop. We report results of three tests.

4.3.1 Stepped Inputs: 0 to 100%

In Figure 4.5, all the input rates are lock-stepped up and down from 0 to 100%, a worst case for response delay and stability. As we can see from figure, it takes about

450 cycles to reach the input speed. When the scheduler rate crosses the 50% bar, it suddenly goes to zero and then it gets back on track after a few clock cycles. This short pause is necessary to change the voltage. Everything gets stalled through clock-gating and resumed only when the supply voltage has settled.

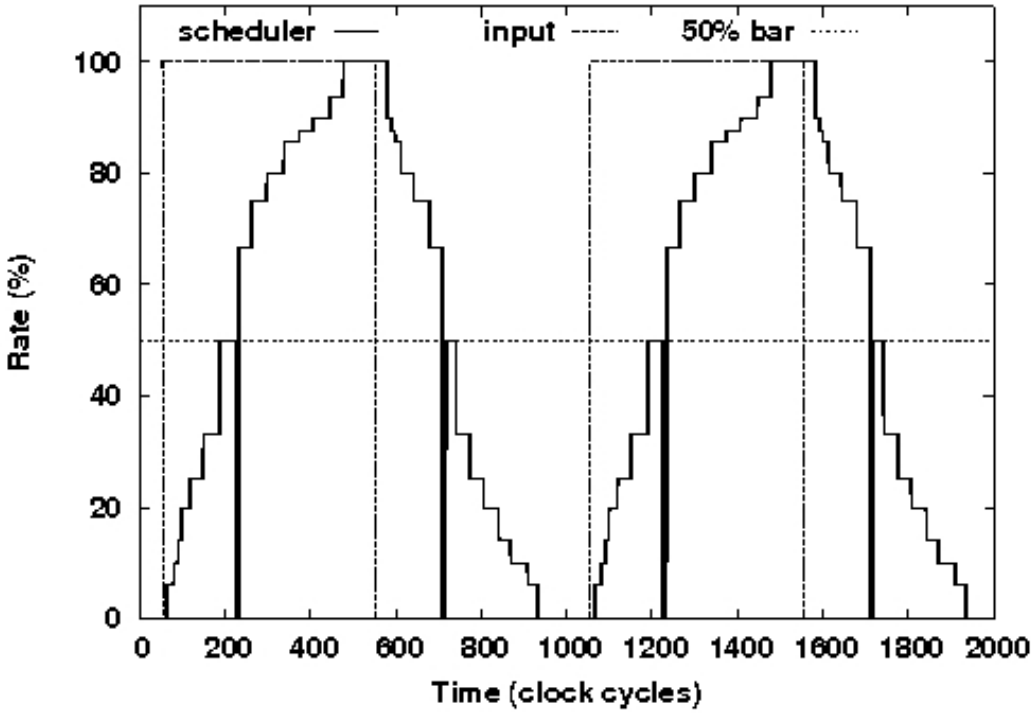

FIGURE 4.5: Response to Lock Stepped Input from 0 to 100%

4.3.2 Stepped Inputs: 0 to 33%

In Figure 4.6 inputs are stepped from 0 to 33%. Since the 50% bar is not crossed, there is no need to change the voltage and so to fully gate the clock. The figure shows an overcorrection of the feedback loop that determines a little overshoot in scheduler rate before settling to the correct value. The smaller step requires less clock cycles to get on track, around 100.

4.3.3 Approximately Sinusoidal Inputs

In Figure 4.7, one of the input rates, input 0, goes from zero to 100% and then back to zero, creating a roughly sinusoidal stimulus for the control loop. The other four input rates, inputs 1-4, are kept constant and equal to 33%. The scheduler tracks the maximum rate. Therefore, when input 0's rate rises above 33%, the scheduler rate tracks it fairly well. When input 0's falls below 33%, the scheduler tracks the maximum rate, which is

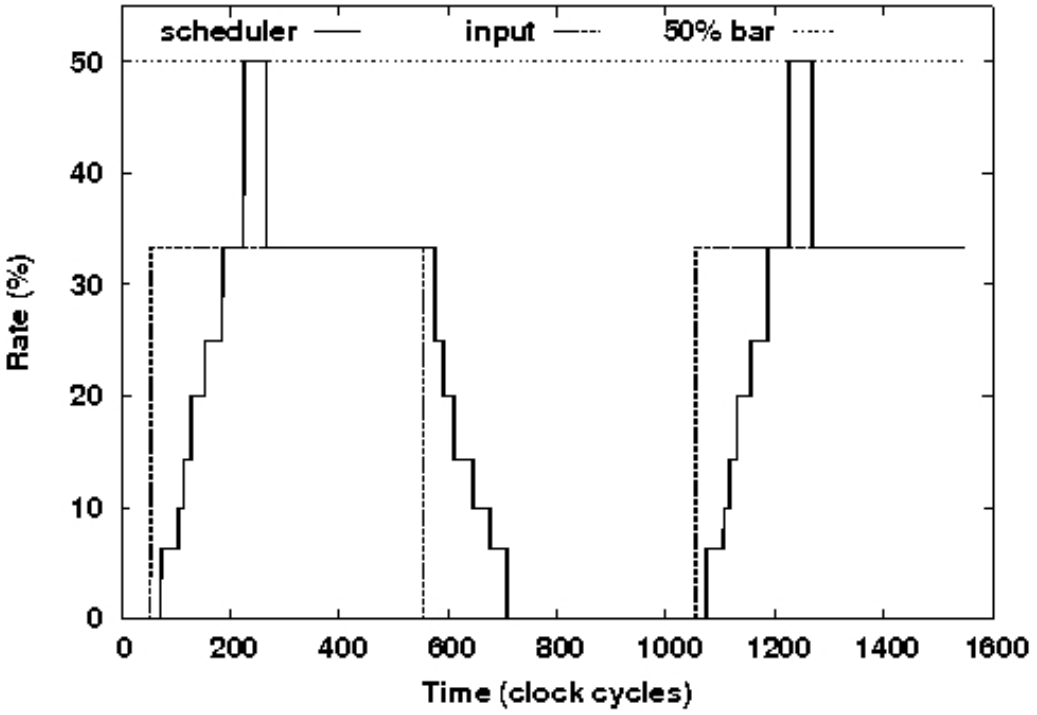

FIGURE 4.6: Response to Lock Stepped Input from 0 to 30%

33% in that situation, and produces soon a constant rate. Slower change in input speed makes the system track it responsively and without substantial delay. Again, voltage changes occur in this test, as shown by the abrupt schedule changes clearly visible in figure, when the scheduler rate crosses the 50% bar.

4.4 Implementation

Our target is a 45 nm CMOS technology. The switch has been synthesized for the most part, using a standard-cell library and Synopsys Design Compiler. We performed the logic synthesis with a library of cells characterized at the highest voltage, 1.15 V. We obtained a maximum clock frequency of 0.93 GHz (1.08 ns). The area of the circuit is 56000 square micrometers. The area overhead of the control loop and the scheduler is only 2.8%.

We manually designed instead the voltage conversion circuits and the driver circuit for the power pMOS transistors. To determine the lower voltage Vlow , we plot the critical path delay of the switch as a function of supply voltage. We choose Vlow as the voltage that doubles the delay (2.16 ns, and so halves the clock frequency) with respect to the value at Vhigh. Figure 4.8 shows that this value is 0.89 V for our circuit and our technology.

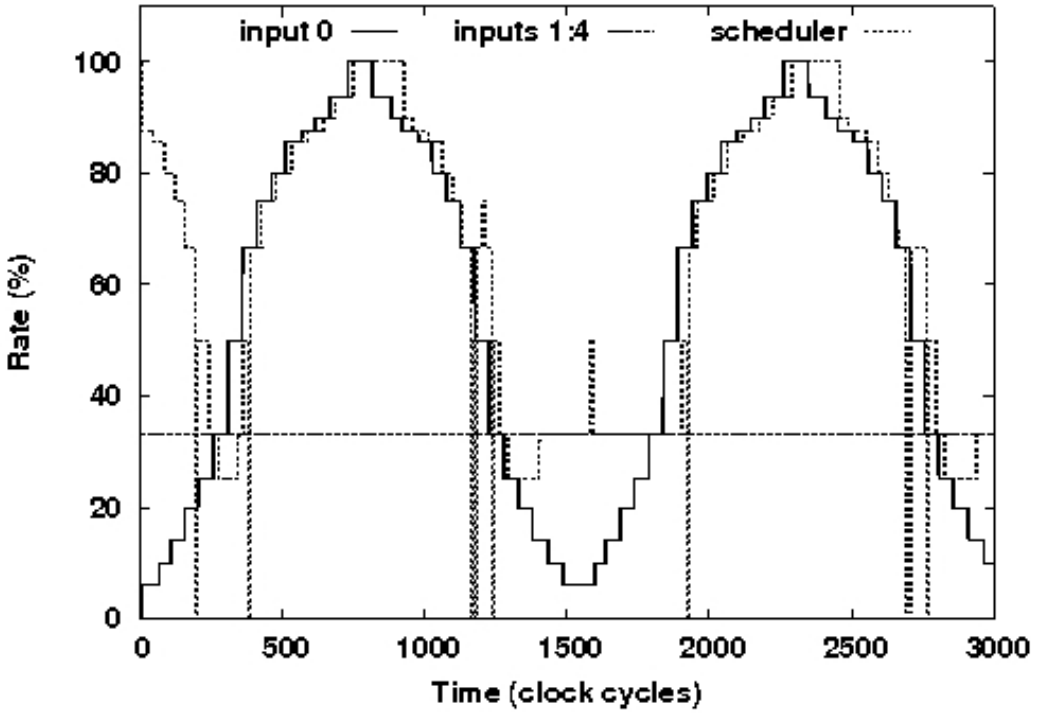

FIGURE 4.7: Response to Linearly Variable Input

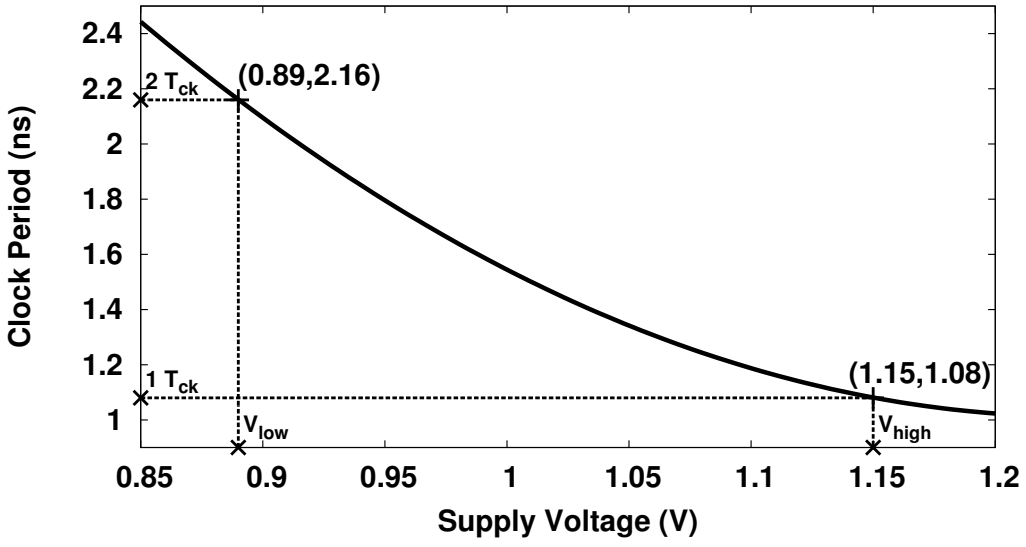

FIGURE 4.8: Minimum switch's clock period as a function of supply voltage

The circuit in Figure 4.9 provides the scaled supply voltage to the switch. We use standard cell gates from our 45 nm library, except for the two pMOSs, which have low threshold voltage, whereas standard-cell gates' transistors have standard threshold voltage. PMOSs' size (W/L=10um/0.04um) guarantees less than 0.5% voltage drop at both Vhigh and Vlow.

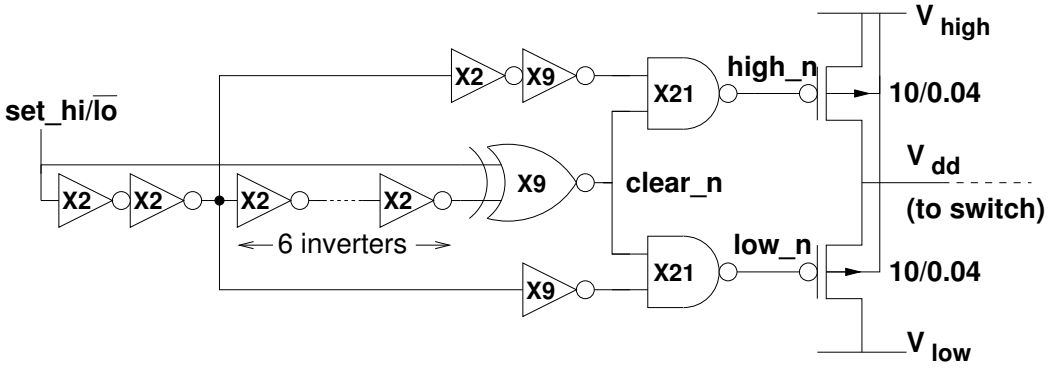

FIGURE 4.9: Voltage scaling driver circuit

Gate labels in Figure 4.9 indicate gate strength (X2, X9, etc.). The power supply for these gates is always Vhigh. Inverters serve as delay lines and as buffers to drive the higher load of nand gates. When a voltage change is requested, the xnor gate and the three-inverter delay line avoid a simultaneous application, albeit temporary, of high_n and low_n commands, by generating a short negative pulse on clear_n signal, which force both high_n and low_n at Vhigh . Waveforms in Figure 4.10 report an example of Spice simulation results and shows the relevant signals involved in a voltage change.

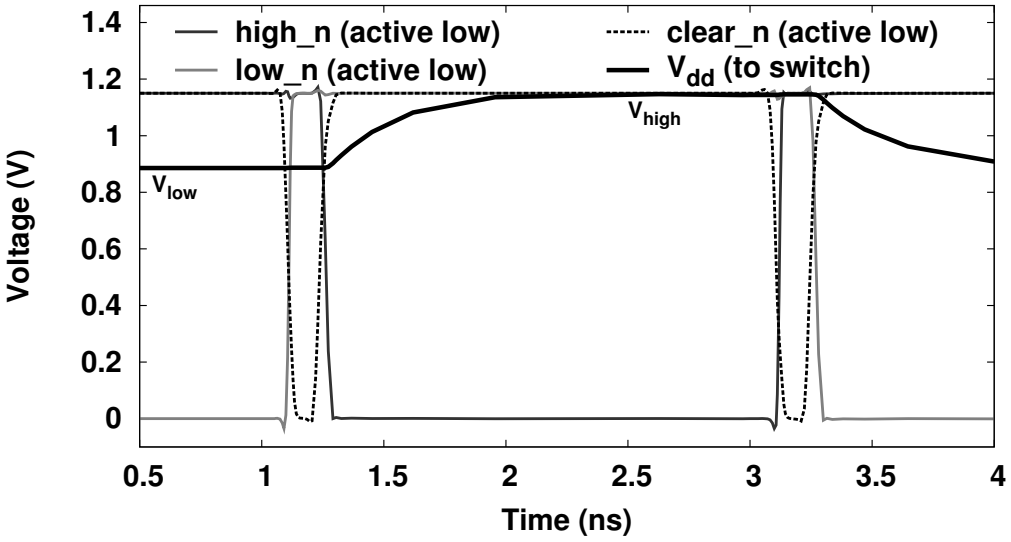

FIGURE 4.10: Signals of circuit in Fig 4.9 involved in a voltage change

Any voltage change takes about 2 cycles of the master clock and consumes approximately 50 fJ. As an example, changing voltage every 100 clock cycles results in an additional

consumption of 0.5 μW, a negligible fraction of the total switch power. The area of the circuit in Figure 4.9, including the two power MOSFETs, is only 25 μm^2 , 0.04% of the total area.

Figure 4.11 shows the link between two switches (TX and RX) and the main signals involved in data transmission. Whenever the TX switch issues a valid flit, the Valid TX signal is set to high for a clock cycle, and a strobe signal is generated at the RX switch (via clock gating) for writing data in the input FIFO (signal FIFO gated Ck in Figure 4.11). The voltage conversion stage is adapted from [24].

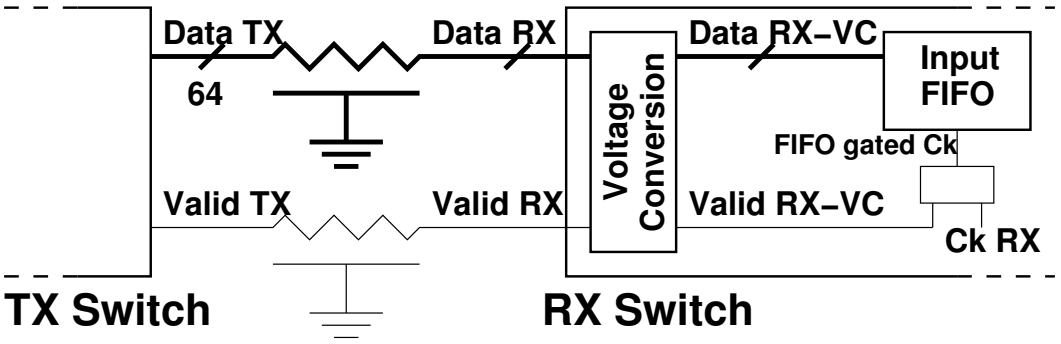

FIGURE 4.11: Simplified circuit for switch-to-switch data transmission

Waveforms in Figure 4.12 report post-synthesis Spice simulations of signals in Figure 4.11 TX Switch, clocked at 1/3 of maximum frequency, transmits data at 33% rate and at 0.89 V. Data, validated once every three cycles with Valid TX signal, are sent through a 2-mm link (modeled in Spice as a RC distributed line, with parameters taken from our 45 nm technology file). The RX switch is working at a higher speed, and therefore its voltage is 1.15 V. As shown in Figure 4.12, data and valid signals are converted to the higher voltage. Valid TX is used at the input FIFO to create a write pulse via clock-gating of the master clock, as shown by signal FIFO gated clock at the bottom of Figure 4.12, which also shows that the FIFO is updated with the newly written data (waveform Data Stored in FIFO).

4.5 Switch Power Evaluation

Power evaluation for the synthesized part of the switch consisted in running a 200 μs simulation at gate-level (post-synthesis) with random input patterns and back-annotation of net activities into Synopsys Design Compiler.

Curves No Scaling and F-Scaling in Figure 4.13 report switch power versus input rate with no DVFS (highest frequency and voltage), and with frequency scaling (variable frequency, highest voltage), respectively. In a steady state condition, the input rate

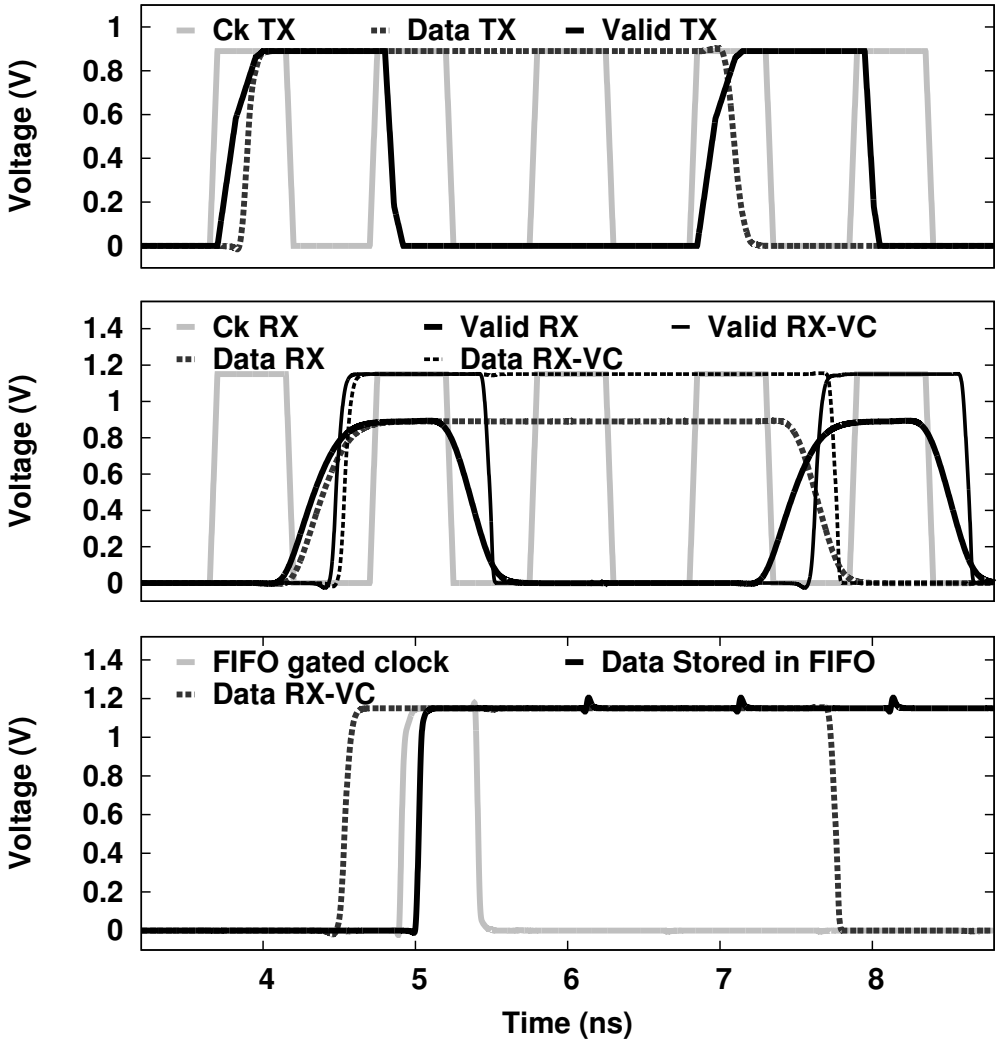

FIGURE 4.12: Simulation of switch-to-switch data transmission

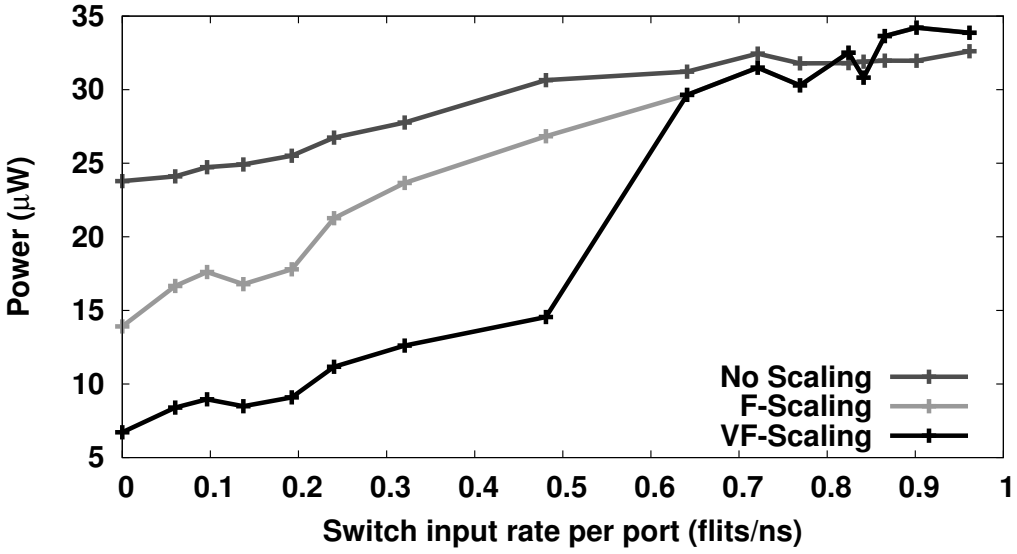

FIGURE 4.13: Power vs. frequency curves for a single switch

is matched to the calculated schedule rate. At half frequency range, power reduction of frequency scaling is 14%. VF-Scaling curve shows that DVFS with two voltages produces a relevant power saving: a further 2X reduction in power is possible when using the dynamic voltage scaling together with frequency scaling. These results are confirmed in the next Chapter by network-level results.

The power consumed by the controller is a very small fraction of the total power, around 3.3%. We can claim that the DVFS mechanism that we propose is very cost efficient, both area and power-wise.

4.6 Summary

We have discussed a switch for Networks-on-Chip with a simple dynamic voltage and frequency controller which does not use any DC-DC converter, PLL or DLL. Instead, we use two voltages to choose from for voltage scaling and a simple clock scheduler for effective frequency scaling. We discussed how the switch autonomously chooses frequency and voltage to keep up with the input rate through a digital control feedback loop. The controller uses a very small fraction of total switch power and area, as we confirmed through synthesis results obtained targeting a 45 nm CMOS technology. Results show an advantage of the dynamic voltage scaling method over a simple frequency scaling policy up to 2X.

Chapter 5

Local Automatic Rate Adjustment in Network-on-Chips with a Simple DVFS

In this Chapter we extend the results of the previous chapter, which focused on a single switch, by reporting global results of power consumption obtained at the network level. We give our NoC the name of LAURA-NoC, which stands for Local Automatic Rate Adjustment Network-on-Chip. The reason is that each switch determines locally, without a global controller, the best clock frequency and voltage, using the DVFS controller and the feedback loop that were reported in the previous chapter. To evaluate the power consumption of LAURA-NoC, we considered both the cases of synthetic traffic traces and realistic traffic generated by common video applications. We evaluated power using Synopsys Design Compiler and SAIF backannotation from accurate post-synthesis gate-level simulations.

5.1 Power Performance of LAURA-NoC with synthetic traffic

As an example of how a local approach to DVFS in a 16-switch NoC outperforms a global one, let us consider first the synthetic traffic scenario in Fig. 5.1. Here, load is 100% for four switches and 50% for the other twelve switches. In a global approach to voltage and frequency scaling, a unique voltage and a unique frequency are set for the entire NoC. The network bottleneck, which is the 100% load of four switches, forces voltage and frequency to take the highest value, even though many switches can run at

a 50% smaller frequency, and so at a lower voltage. As the bar diagram shows, in this scenario there is no saving compared to a no-DVFS case tuned to the worst-case traffic rates.

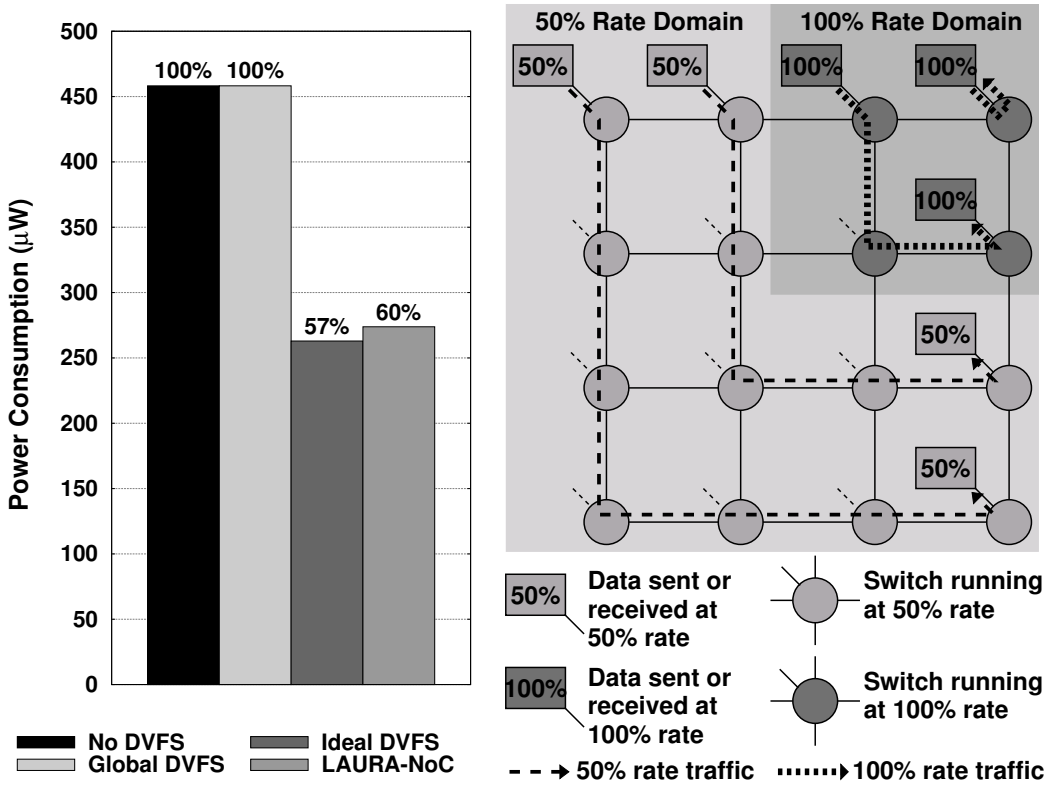

FIGURE 5.1: Local versus global DVFS in an NoC with synthetic traffic. The local approach tracks better the local traffic by controlling voltage and frequency of individual switches.

The example of Fig. 5.1 is a situation of unbalanced traffic. The more diverse and unbalanced the traffic is, the better the LAURA-NoC performance. Let us now consider other scenarios in which we vary the traffic balance in the network. These scenarios are represented by the traffic patterns in Fig. 5.2(a)-(c).

In Fig. 5.2(a), fourteen switches are idle, whereas the traffic between the remaining twos varies between 0 and 100%. This is the most favorable case for LAURA-NoC, especially for high x rates. In Fig. 5.2(b), the lower diagonal part of the mesh runs at 50% rate, whereas the upper diagonal has a variable rate between 0 and 100%. This is a less favorable situation. Finally, Fig. 5.2(c) is the worst case for LAURA-NoC. The left half of the network nodes inject a 50%-rate traffic, while the right half injects a traffic at rate x between 0 and 100%. Especially when x>50%, many of the switches, also those on the left half, are heavily loaded. Therefore, the clock frequency must be high to keep up with the traffic rate. We expect an advantage only for x<50%.

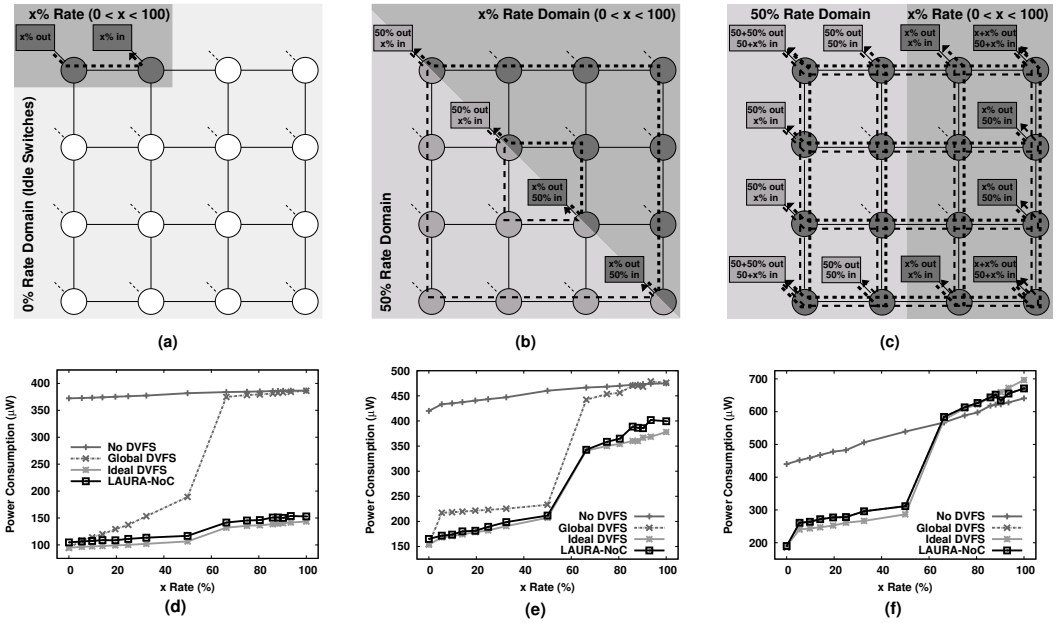

FIGURE 5.2: NoC power performance for synthetic traffic

For each scenario in Figs. 5.2(a)-(c), we report a corresponding graph in Figs. 5.2(d)-(f), with four power curves as a function of x: 1) no DVFS, 2) global DVFS with two voltages, 3) LAURA-NoC, and 4) ideal local DVFS with pre-computed frequency and voltage per switch.

The difference between ideal and LAURA-NoC is always within a few percent, because the overhead of local rate estimation and voltage actuation is small. The no-DVFS curves show a little scaling of power with rate, because of the variation of activity in switches caused by traffic load changes.

In Fig. 5.2(d), LAURA-NoC's power is 2.5 times lower at x=100% than no-DVFS and global DVFS. As x decreases, the advantage with respect to a no DVFS case grows, reaching 3.75 times at x=0, while the advantage with respect to global DVFS fades out, especially below 50% rate because of the sudden voltage change (high to low). At x=50%, LAURA-NoC is still ahead of global DVFS by 1.5x.

In Fig. 5.2(e), LAURA-NoC's power remains consistently below the global DVFS curve. The maximum advantage is at x=66%, where saving reaches 29%. At full rate (x=100%) the power saving is 19%.

Fig. 5.2(f) is where LAURA-NoC has no advantage with respect to global DVFS and the two curves almost overlap. At a very high rate, the traffic requires very high clock frequency to keep up with the load and therefore the no-DVFS case, which has no overhead whatsoever, actually performs better. But as soon as x falls below 66%, the DVFS curves show the advantage with respect to a no-DVFS policy.

5.2 Power Performance of LAURA-NoC with realistic traffic

We report results related to four relevant video processing applications, already used in the context of NoCs [25]: Picture-in-Picture (PIP), Multi-Window Display (MWD), Video Object Plane Decoder (VOPD), and MPEG4. Graphs in Figure 5.3(a) and Figure 5.4(a)-(c) show the normalized communication rates between the application tasks. NoC arrangements in Figure 5.3(b) and Figure 5.4(d)-(f) show how tasks have been mapped and show the resulting normalized traffic load of the NoC switches.

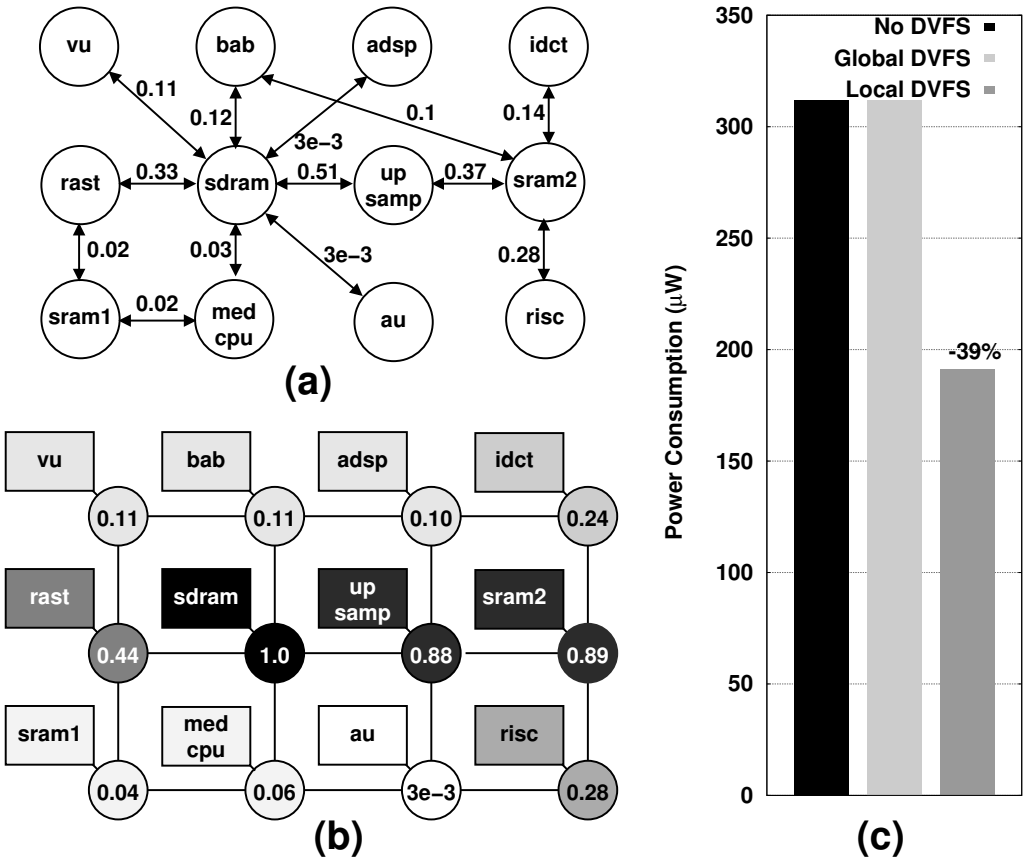

FIGURE 5.3: Local versus global DVFS in an NoC. (a) Communication graph in MPEG4 annotated with normalized rates. (b) MPEG4 tasks mapped to a 12-switches NoC, gray shading proportional to normalized switch rates. (c) Power consumption: The local approach of LAURA-NoC tracks better the local traffic by controlling voltage and frequency of individual switches.

We compared LAURA-NOC with two other NoCs. The first one uses a global DVFS approach. We remove the local rate estimation and assume that a global controller can set any voltage in range 0.75 V-1.25 V (for DVFS in 45 nm technology [2]) and set the corresponding frequency for the whole network (clock period obtained with curve in Figure 4.8. The second one uses an ideal distributed DVFS: Each switch has a local

controller that selects voltage and frequency in the same range of the global one, and switches communicate with small dual-clock FIFOs and voltage level shifters [1].

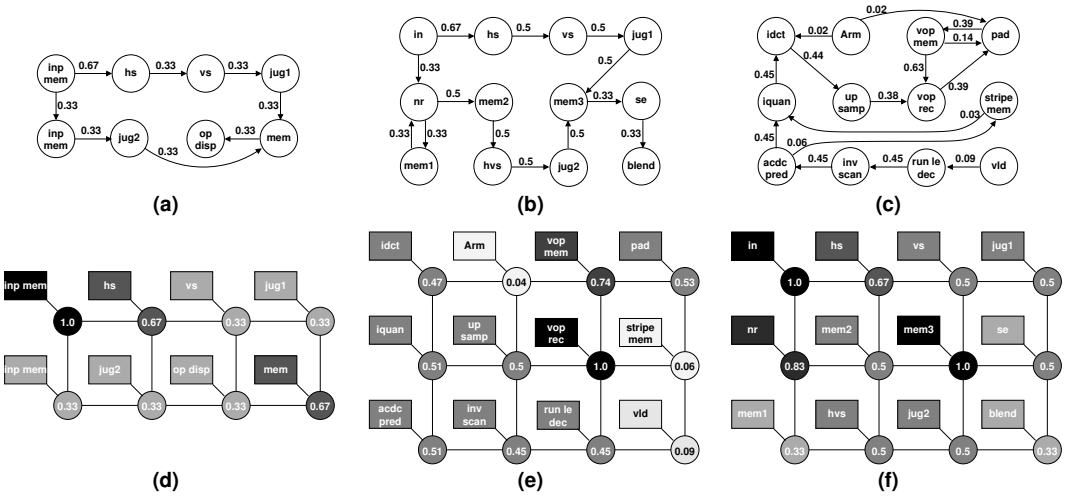

FIGURE 5.4: (a)-(c) Communication graphs annotated with normalized rates and (d)-(f) NoC mappings annotated with switch load rates for three video applications:(a),(d) PIP, (b),(e) MWD, (c),(f) VOPD (MPEG4 is in Fig.5.3).

Figure 5.5 shows that the advantage of our approach compared to the global one is clear, with an average power saving of 33% over the four video applications. The ideal distributed DVFS case is on average 18% better than LAURA-NoC, but we did not include in this evaluation the power of the individual DC-DC controller and PLL. We expect these overheads to be even more than such 18% advantage.

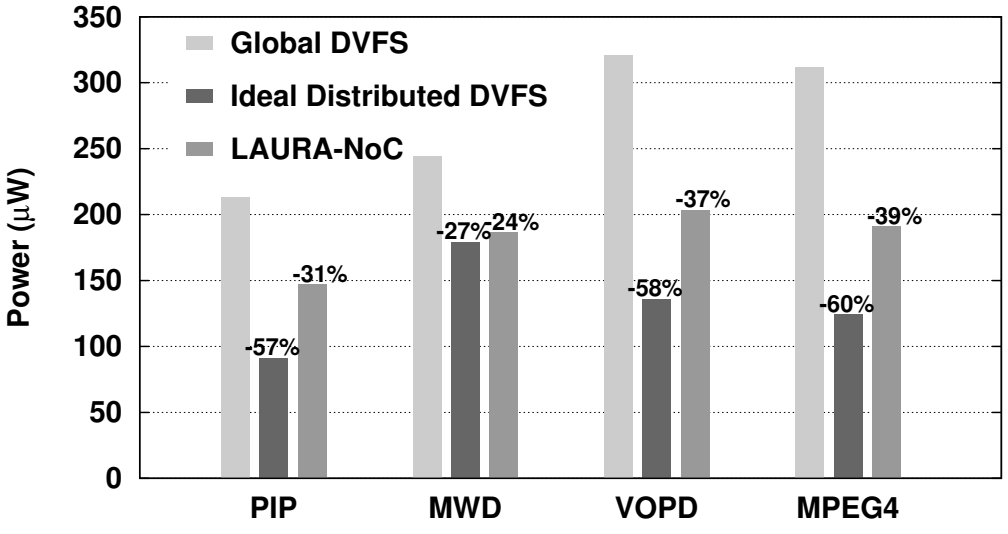

FIGURE 5.5: NoC power for PIP, MWD, VOPD, and MPEG4 video applications

[1]Incidentally, this is how any DVFS NoC connects to computing or memory nodes with different voltage and frequency, no matter the DVFS strategy.

In terms of performance, the three NoCs exhibit different latency, while guaranteeing the throughput required by the four video applications. Figure 5.6 reports average, minimum, and maximum latency measured when NoCs are loaded with the traffic generated by these applications. The ideal DVFS approach pays the lowest power consumption with the highest average and variance of latency. As pointed out in [20], the reason is the additional latency for crossing different clock domains. LAURA-NoC uses a single, global master clock and thus has limited latency overhead.

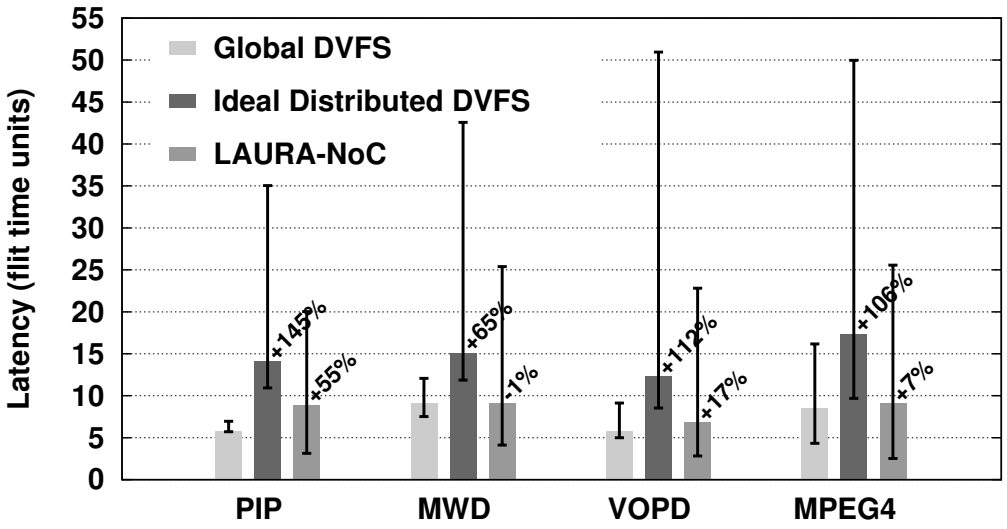

FIGURE 5.6: NoC latency for PIP, MWD, VOPD, and MPEG4 video applications

5.3 Related Work

DVFS in NoCs is a relatively new field. A global approach was used in Intel's single-chip cloud computer (SCC) [2], and was proposed in [20] because it avoids resynchronization. Distributed approaches with resynchronizing FIFOs were proposed in [26] [27] [28] [22]. LAURA-NoC avoids resynchronization by construction. Fully-asynchronous approaches to NoC with distributed DVFS are reported in [1] [29]. LAURA-NoC complies instead with a standard synchronous paradigm that permits to use commercial EDA tools for design and verification.

Some authors propose a full-fledged DC-DC converter per switch [28] [30] [31]. As noted in [22], area and power overhead makes this approach impractical for NoCs. LAURA-NoC uses instead a simpler voltage scaling method and has commonalities with [1] [31] [22]. Our voltage controller, however, is less complex than in [1][31] because the current load is small enough for a simple transistor-based two-voltage switch. Links in [1] always remain at a high voltage. LAURA-NoC's links use the same voltage, low or high, of the switch they depart from. There are no implementation details in [22] to

permit a comparison. A similar circuit was used in [8] for the cores that communicate through an NoC, whereas the NoC switches did not use DVFS. The work in [8] confirms our results that switching between two voltages can be very fast.

Methods to estimate the rate in an NoC have been reported in almost all previously mentioned papers, and virtually all of them refer to seminal paper [32], which reports a general method applicable to on-chip and off-chip networks. In [32], metrics like buffer or link utilization were used for rate estimation and to control dynamically voltage and frequency of communication links. Likewise, [27] focuses on NoC links, but on-chip links are not the main source of power, and thus DVFS has to scale voltage and frequency also of the switches.

LAURA-NoC uses input buffer usage as it performs better than link utilization [32][29]. In [28] and [22], buffer usage is compared to thresholds to decide whether voltage and frequency have to change. In [1], the NoC node is powered up as soon as a message enters it and powered down when the node gets empty. These approaches do not filter rapid traffic rate fluctuations, whereas LAURA-NoC uses a time-averaged buffer utilization. Averaged buffer utilization is also used in [30], but an impractical DVFS controller is assumed.

In [29], comparisons between average buffer usage and thresholds determine voltage and frequency out of three values. We follow instead a more control-theoretical approach. In this LAURA-NoC has features in common with [26] and [20], which however use a global controller. Our feedback-loop mechanism takes inspiration from [33], in which a similar approach is used for a processor's DVFS controller.

Chapter 6

Reduction of Leakage Power in Networks-on-Chip with Buffers Power-Gating

6.1 Introduction and background

We previously focused on reducing a NoC's dynamic power via DVFS. Static power is, however, as important as dynamic power in current CMOS technologies. Therefore, here we discuss our work on reducing a NoC's leakage power. In particular, we focus on the higher source of leakage in a NoC switch like the one in Figure 6.1, which is represented by the SRAM buffers that make up the input FIFO queues. Since we do not want to impact performance, and given that FIFO buffers are critical for a NoC's performance, we focus on reducing power of SRAM buffers that are not fully utilized. NoC traffic may present large spatial and temporal variations, and so not fully loaded NoC nodes can be reconfigured to save power. We assume that the SRAM buffers can be partitioned in banks, as shown in the layout in Figure 6.2, and that each bank can be individually put in a low-power state by cutting its power supply via power MOSFETs as shown in Figure 6.4, a technique termed power-gating (PG) [34].

PG in NoCs has been previously explored by researchers, but we make original contributions to the state of the art.

In [35] only leakage saving and NoC performance are reported, whereas aspects such as area and energy overhead of power-gating and impact of power-gating on clock frequency, which we cover in this thesis, are not discussed.

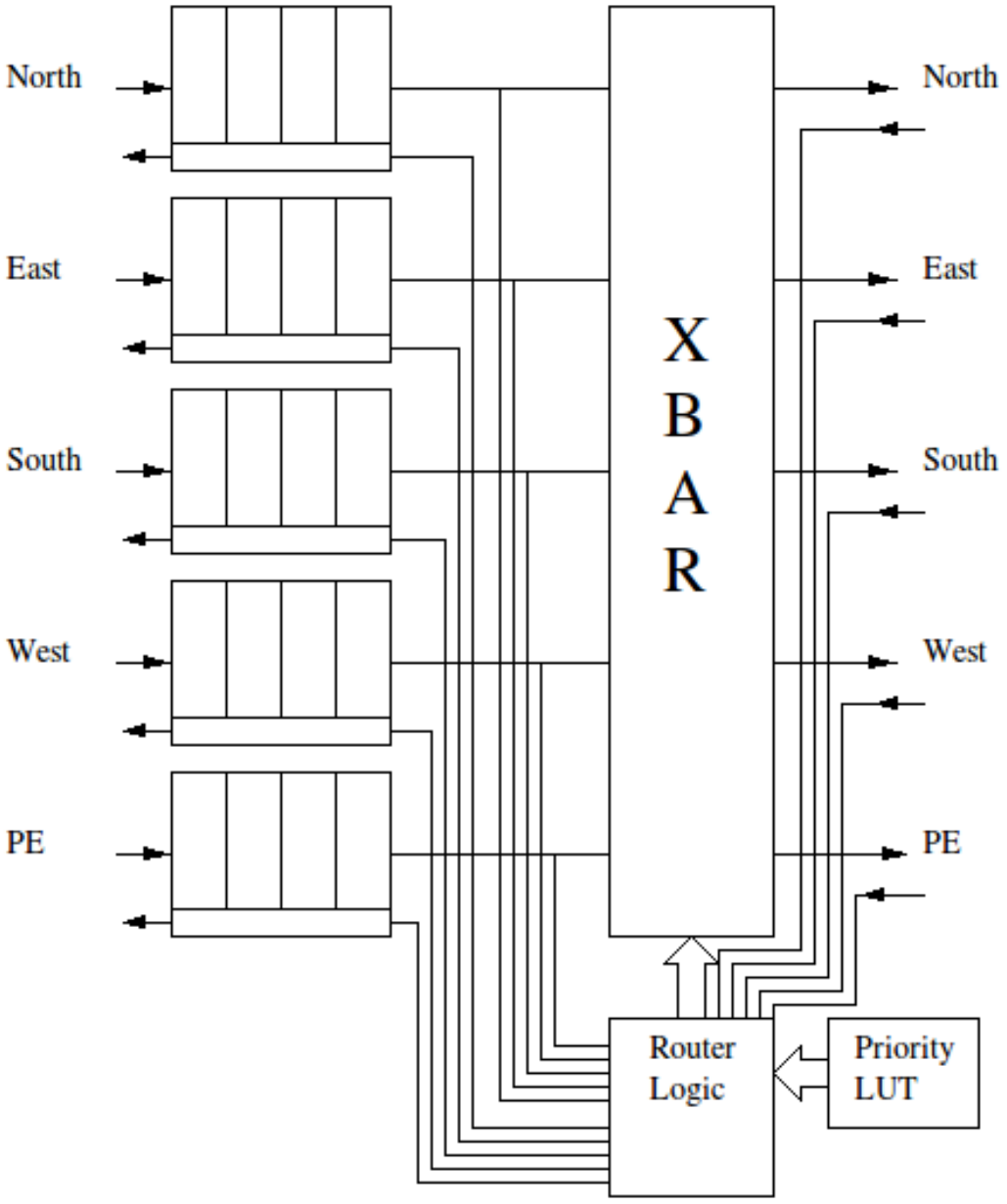

FIGURE 6.1: A NoC Switch

This latter aspect is discussed in [18], but power-gating is used aggressively to completely turn-off a buffer. This choice minimizes leakage power but may negatively affect performance, as flits arriving at a switch's inputs when the buffer is off have to be pushed back and delayed. We avoid this problem by keeping one portion of the buffer always on.

The work in [36] proposes to control each buffer entry via power-gating but does not quantify the overhead of such approach. Moreover, the conventional credit-based flow control has to be modified, a *congestion* detection circuit added to the router logic, and a congestion signal added to the links connecting switches together.

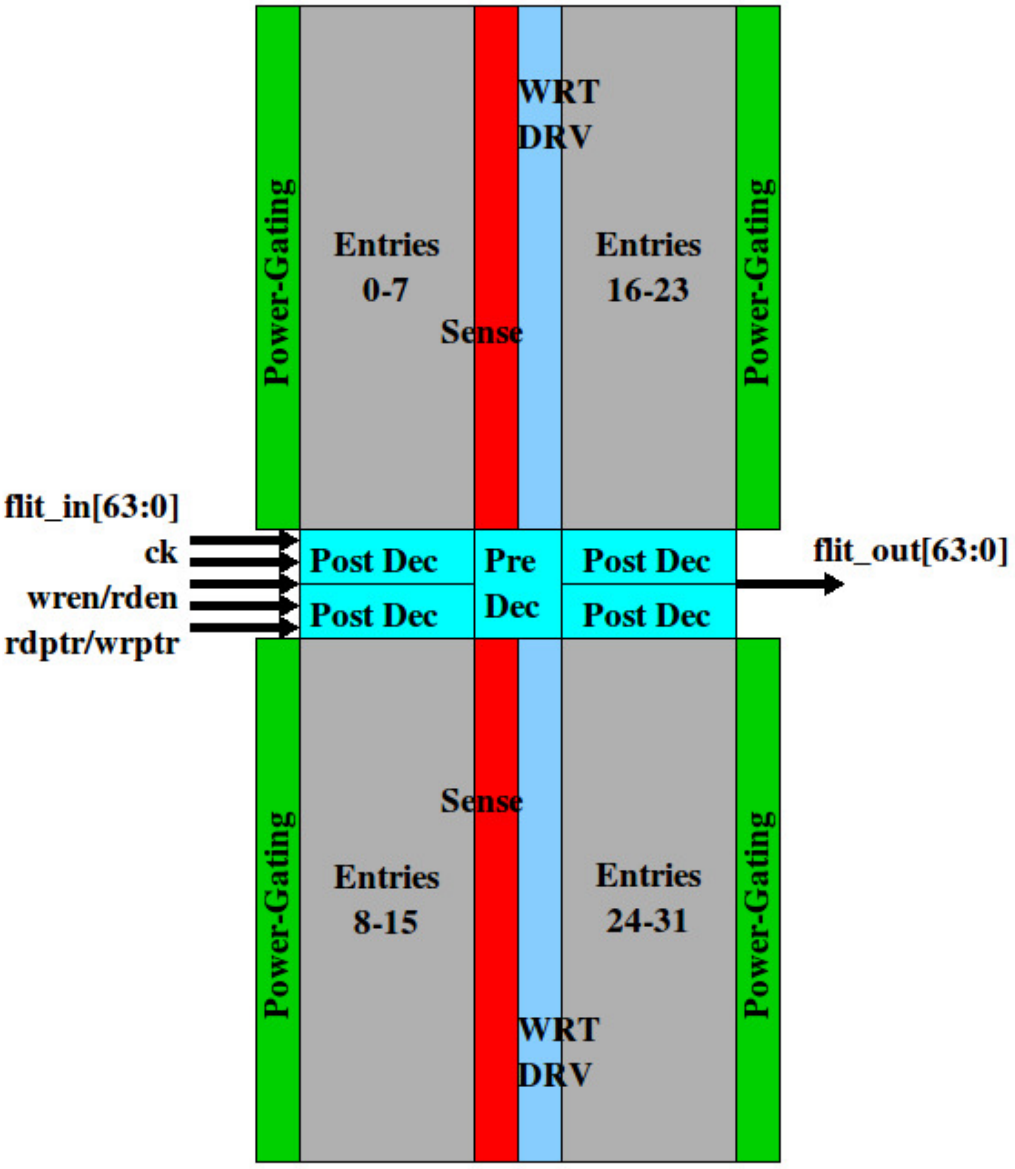

FIGURE 6.2: Layout of 4-banks input FIFO SRAM buffer with per-bank power gating

In [37] authors evaluate various power-gating policies, which require special wake-up signals added to the links. Moreover, their power-gating method is based on a modified standard-cell library of CMOS gates, whereas we use a regular library. They evaluate overhead of power-gating on power, but not the impact on clock frequency, which instead we do.

6.2 NoC with Buffer Power Gating

To determine whether a buffer can be power-gated or not, we keep track of its average utilization U via a simple cumulative moving average circuit. When U falls below a given

threshold for power-down, U < THpd, we keep active only one bank and put into sleep-state all other banks via power-gating. We keep on updating the average utilization of the active bank. When U rises above a threshold for power-up, U > THpu, we wake all banks up from sleep-state, and the SRAM becomes fully available. To avoid oscillations, we set THpu > THpd, which creates a hysteresis.

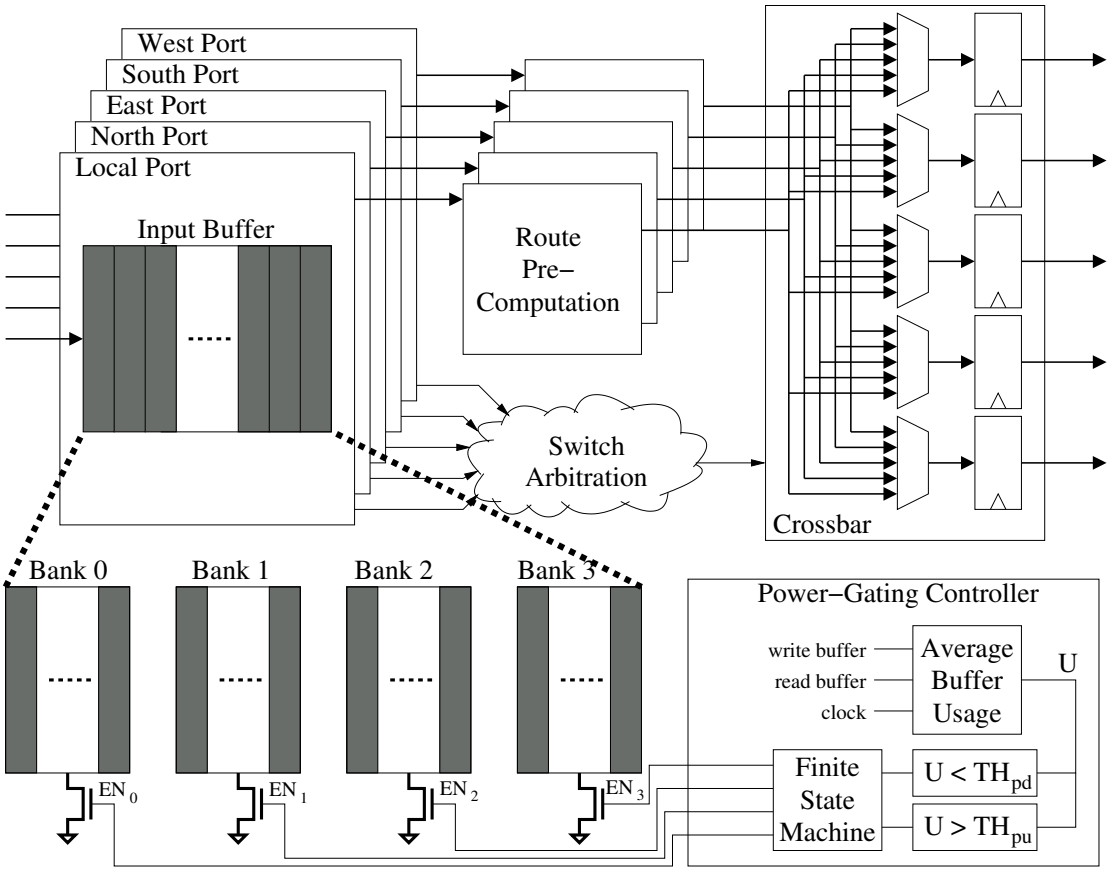

FIGURE 6.3: Microarchitecture of the switch, with detail of the power-gating technique applied to the input buffers partitioned in banks.

Figure 6.3 shows the switch microarchitecture and shows that the PG controller consists of a block that keeps track of the buffer utilization (this is where the cumulative moving average is computed), two comparators to determine if the utilization is above or below the thresholds, and a finite state machine (FSM) whose outputs are the enable signals for the buffers' banks. The FSM makes sure that power-gating is applied only after all flits stored in those banks that are going to be turned-off have been removed from the buffer and delivered at the switch output ports.

The controller belongs to the class of on-off controller with hysteresis (bang-bang controller). If a single threshold is used, a much more frequent on-off switching occurs when buffer utilization has little variations above and below the threshold.

The size of the NMOS footers is a critical design parameter. A small size corresponds to less gate capacitance and so to faster on-off switching and less energy. But it also entails a higher V_{DS} voltage drop when the bank is active; the consequence is a lower supply voltage for the bank ($V_{dd} - V_{DS}$) and so a longer propagation delay of logic gates and access time of memory cells inside the bank. To determine the best size of the NMOS transistors, we ran Spice simulations with the post-synthesis netlist of the switch, to which we added the NMOS footers. In our 1.15 V 45 nm technology, the switch's critical path dictates a clock frequency of 1 GHz (at T = 25°C, nominal process). We set the NMOS size such that the critical path delay increases by no more than 1%. In our technology, this corresponds to a 126-μm MOS channel width. The energy required per switch is 0.67 pJ. We show next that this overhead is negligible compared to power saving.

The turn-on delay of a bank also depends on the NMOS size and in this case is about 3 ns, which corresponds to 3 cycles at 1 GHz. To account for variations (e.g. process, voltage, and temperature), the controller waits 4 cycles before enabling read/write operations in the turned-on banks. To limit the impact of turn-on delay, the active bank must not be full when the other banks are being activated, so that it can absorb flits arriving in the meantime. This problem is solved with a power-up threshold TH_{pu} sufficiently smaller than the bank size.

Here lies the trade-off between power that can be saved and performance. The smaller the size of the active bank, the larger the power that can be saved, but also the higher the probability of having full buffers when traffic changes occur. Full buffers mean higher network congestion and so longer delivery time of NoC packets.

To evaluate this trade-off, we built a NoC made of wormhole switches adapted from the one presented in previous chapters and from [16], with the addition of power-gating in input buffers and the PG controller. We designed the NoC switches in a 45-nm CMOS technology. Here we report results obtained after evaluation of power with accurate back-annotation of NoC traffic from gate-level simulations. We consider the traffic generated by four realistic video applications – PIP, VOPD, MWD, MPEG4 from [25] – mapped onto an 8-nodes or a 12-node NoC, as shown in Figure 6.5 for the MPEG4 case. We consider three cases of buffer/bank size (in flits, the elementary data unit of NoC packets): 32 flits/8 flits (25%), 16/8 (50%), and 16/4 (25%). Each flit is a 64-bit data. The choice of the thresholds is critical for both power and performance. We obtain the best results in our simulations by setting $TH_{pd} = 1$ and $TH_{pu} = 2.5$. Clock frequency is 1 GHz, supply voltage is 1.15 V, and temperature is set to 25 C.

To understand how much leakage power can be saved, we look at the average utilization of buffers in a typical scenario. Fig. 6.6 illustrates an NoC mesh arrangement of 12

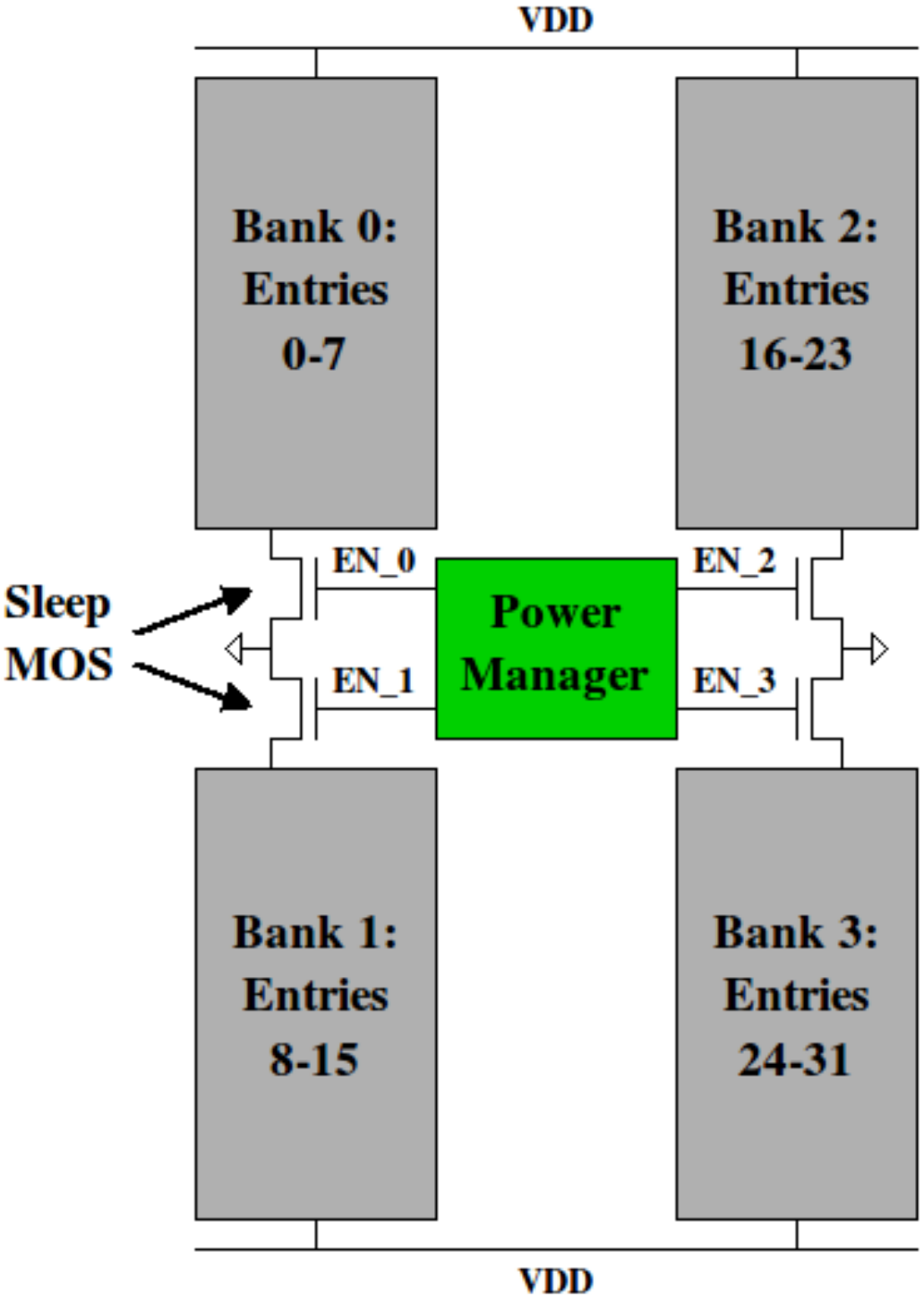

FIGURE 6.4: Power-Gating with sleep MOS footers. In sleep-mode only one bank is active

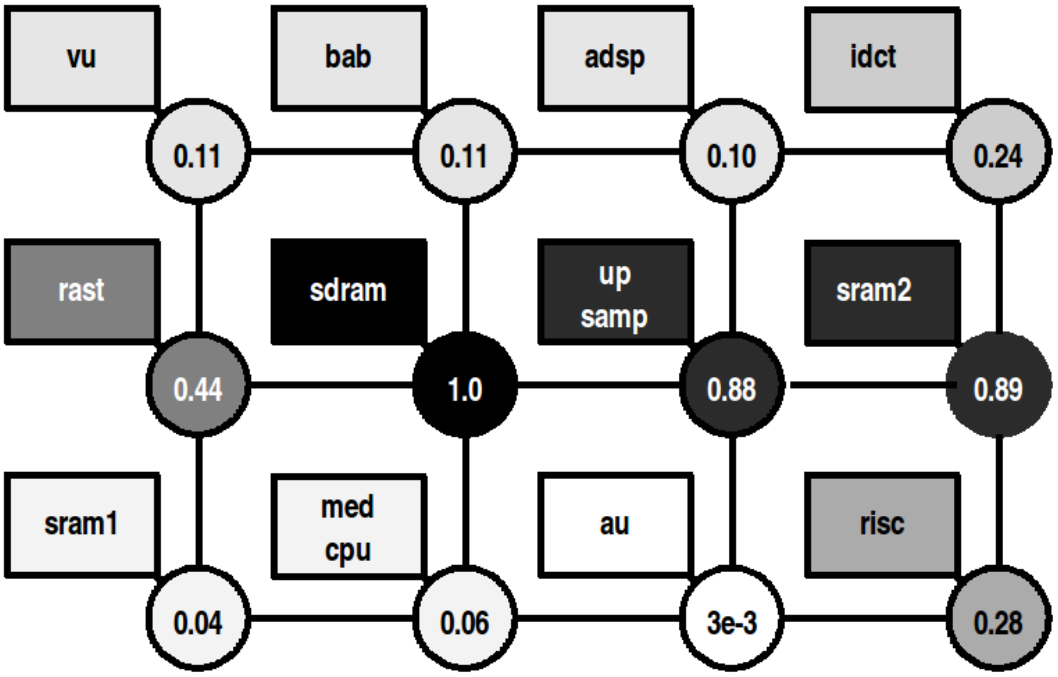

FIGURE 6.5: MPEG4 tasks mapped on the processing elements of a 4x3-switch NoC. Numbers indicate normalized load of the switches. Buffers in highly loaded switches will likely be fully utilized, whereas buffers in other switches can be put in low-power state

switches connected locally to processing elements that execute each a different task of the MPEG4 application. Labels within the switches represent task names [25]. The shading represents traffic load—the darker the higher. Numbers indicate average utilization of input buffers. Only a few buffers have utilization greater than 1%, notably those of the switches with the highest traffic load. Therefore, we can apply power-gating to the majority of the buffers. This observation holds also in the three other video applications that we examined: VOPD (12-switch NoC), MWD (12-switch NoC), and PIP (8-switch NoC).

6.3 Results

The diagram in Figure 6.7 shows that leakage power is about 40-50% of the entire NoC power without buffer power gating (NO PG bars). Leakage can be reduced by about 40% in the 32/8 case (PG bar), leading to a 20% total power saving. In the 16/8 and 16/4 cases, despite a relevant leakage reduction, the overhead of power management (primarily extra dynamic power) cancels the benefit of leakage reduction.

We measured the latency with and without power-gating, reported in Fig. 6.8. We observed a negligible increase of latency in all but the 16/4 buffer/bank size case. We conclude that the size of the active bank must be at least of 8 flits for performance

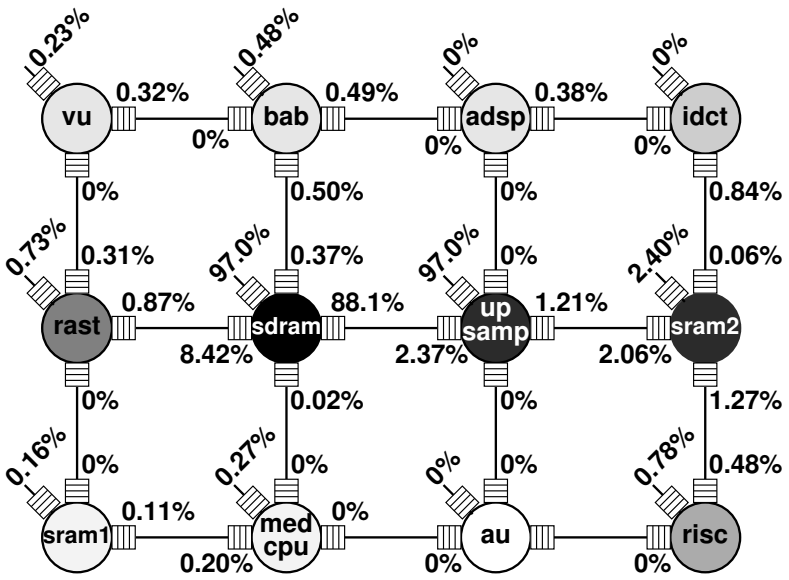

FIGURE 6.6: NoC mesh for MPEG4 application. Shading indicates traffic load of switches. Numbers represent average occupation of input buffers. Labels describe the task performed in each node [25].

reasons. Moreover, the size of the entire buffer must be at least 4 times bigger than the active bank – like in the 32/8 case – to show a substantial power benefit.

Our experiments also show that power switching events do not occur frequently. The distribution of switching events per buffer depends on the application and on the local traffic load. A few switches undergo tens of switching events in 10,000 clock cycles of simulation (up to 61 for a single switch in VOPD) but the average values are only 2.6 events in MPEG4, 2.1 in VOPD, 1.2 in MWD, and 1.0 in PIP. With a 1 GHz clock cycle, and with an energy of 0.67 pJ per switching event, the average power overhead (total energy divided by total time multiplied by the number of NoC switches) is 2.1 μW in MPEG4, 1.7 μW in VOPD, 0.96 μW in MWD, and 0.54 μW in PIP. If we compare these values with leakage in Fig. 6.7, we realize that the overhead is negligible.

6.4 Conclusion

The leakage power in NoC switch buffers can be minimized with power-gating by exploiting their non-uniform utilization. Even when some of the NoC nodes are heavily utilized, others are almost idle. Through a partial power-gating approach it is possible to reduce leakage without incurring a latency penalty. With our approach we reach leakage saving in buffers on the order of 40% with a negligible impact on NoC latency, as measured in realistic video applications.

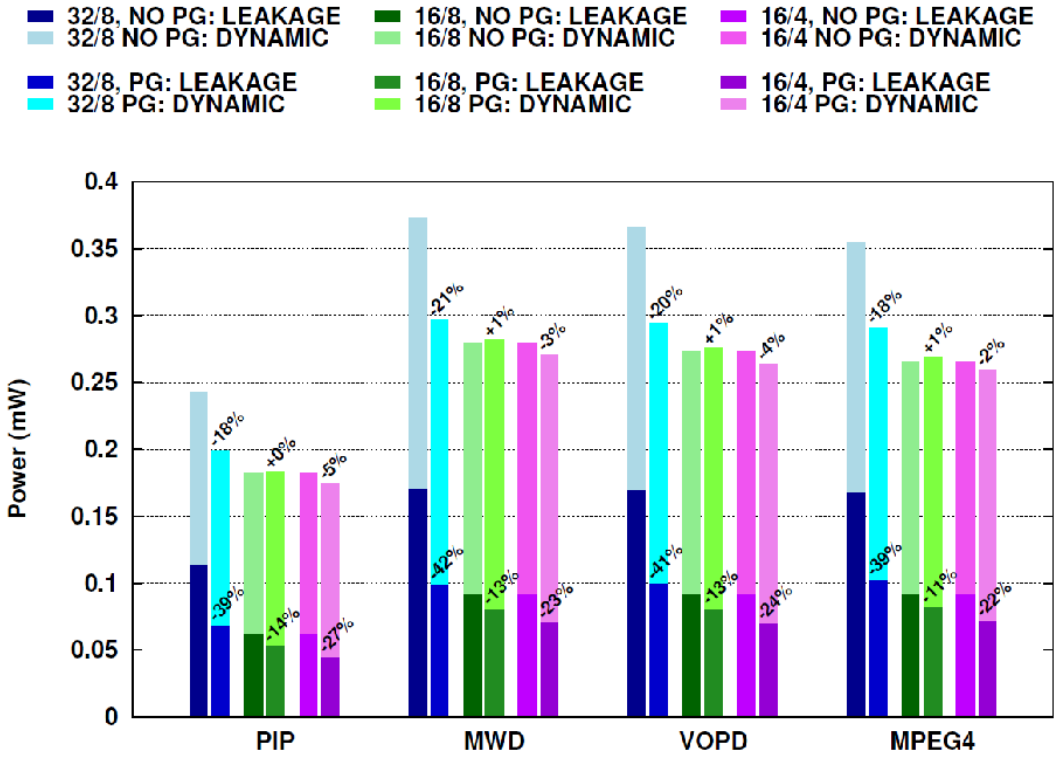

FIGURE 6.7: Power in a 8-switch (PIP) and a 12-switch NoC (MWD, VOPD, MPEG4). Comparison between Power-Gating (PG) and No Power-Gating (NO PG)

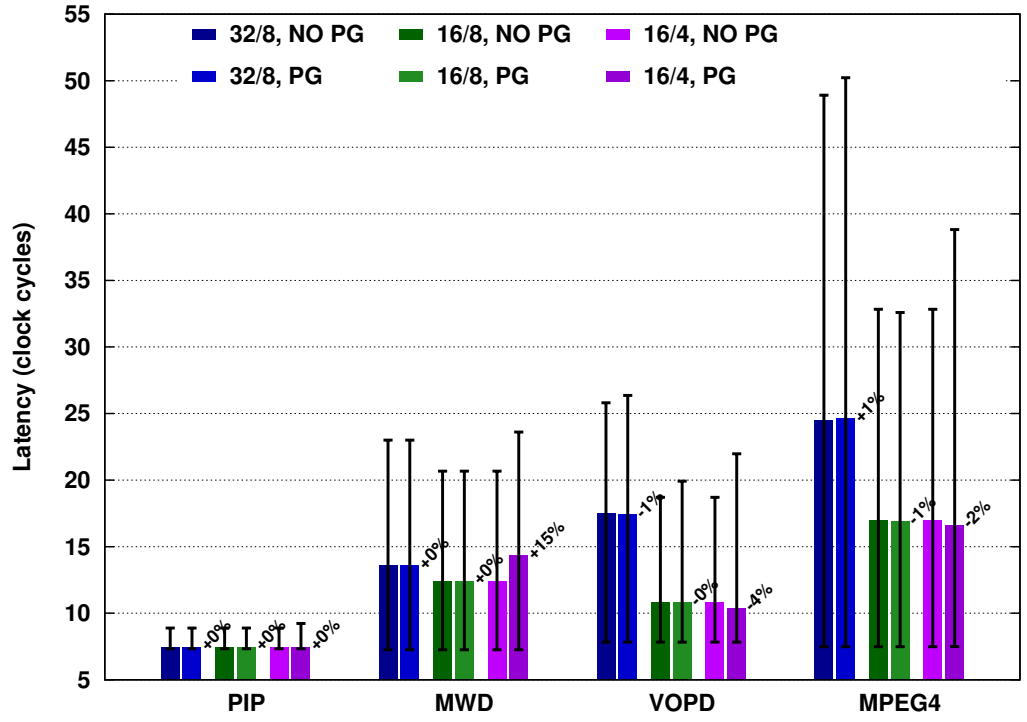

FIGURE 6.8: Latency in a 8-switch (PIP) and a 12-switch NoC (MWD, VOPD, MPEG4). Comparison between Power-Gating (PG) and No Power-Gating (NO PG)

Bibliography

[1] E. Beigne, F. Clermidy, H. Lhermet, S. Miermont, Y. Thonnart, X. T. Tran, A. Valentian, D. Varreau, P. Vivet, X. Popon, and H. Lebreton. An asynchronous power aware and adaptive noc based circuit. *IEEE Journal of Solid-State Circuits*, 44(4):1167–1177, April 2009.

[2] J. Howard et al. A 48-core ia-32 processor in 45 nm cmos using on-die message-passing and dvfs for performance and power scaling. *IEEE Journal of Solid-State Circuits*, 46(1):173–183, jan. 2011. ISSN 0018-9200. doi: 10.1109/JSSC.2010.2079450.

[3] Wonyoung Kim, M. S. Gupta, Gu-Yeon Wei, and D. M. Brooks. System level analysis of fast, per-core dvfs using on-chip switching regulators. In *IEEE 14th International Symposium on High Performance Computer Architecture (HPCA 2008)*, pages 123–134, feb. 2008. doi: 10.1109/HPCA.2008.4658633.

[4] Wonyoung Kim, D. M. Brooks, and Gu-Yeon Wei. A fully-integrated 3-level dc/dc converter for nanosecond-scale dvs with fast shunt regulation. In *2011 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 268–270, feb. 2011. doi: 10.1109/ISSCC.2011.5746313.

[5] B. H. Calhoun and A. P. Chandrakasan. Ultra-dynamic voltage scaling (udvs) using sub-threshold operation and local voltage dithering. *IEEE Journal of Solid-State Circuits*, 41(1):238–245, jan. 2006. ISSN 0018-9200. doi: 10.1109/JSSC.2005.859886.

[6] V. Gutnik and A. P. Chandrakasan. Embedded power supply for low-power dsp. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(4):425–435, dec. 1997. ISSN 1063-8210. doi: 10.1109/92.645069.

[7] H. Kawaguchi, G. Zhang, S. Lee, Y. Shin, and T. Sakurai. A controller lsi for realizing vdd-hopping scheme with off-the-shelf processors and its application to mpeg4 system. *IEICE Transactions on Electronics*, E85-C(2):263–271, feb. 2002. ISSN 0916-8516.

[8] D. N. Truong et al. A 167-processor computational platform in 65 nm cmos. *IEEE Journal of Solid-State Circuits*, 44(4):1130–1144, april 2009. ISSN 0018-9200. doi: 10.1109/JSSC.2009.2013772.

[9] L. F. G. Sarmenta, G. A. Pratt, and S. A. Ward. Rational clocking. In *Proceedings of the 1995 IEEE International Conference on Computer Design (ICCD 1995)*, pages 271 –278, oct 1995. doi: 10.1109/ICCD.1995.528821.

[10] D. G. Messerschmitt. Synchronization in digital system design. *IEEE Journal on Selected Areas in Communications*, 8(8):1404 –1419, oct 1990. ISSN 0733-8716. doi: 10.1109/49.62819.

[11] V. Tiwari et al. Reducing power in high-performance microprocessors. In *Proceedings of the Design Automation Conference (DAC 1998)*, pages 732–737, jun 1998. doi: 10.1109/DAC.1998.136625.

[12] L. P. Carloni, K. L. McMillan, A. Saldanha, and A. L. Sangiovanni-Vincentelli. A methodology for correct-by-construction latency insensitive design. In *IEEE/ACM International Conference on Computer-Aided Design (ICCAD 1999). Digest of Technical Papers*, pages 309–315, 1999. doi: 10.1109/ICCAD.1999.810667.

[13] H. M. Jacobson et al. Synchronous interlocked pipelines. In *Proceedings of the Eighth International Symposium on Asynchronous Circuits and Systems (ASYNC 2002)*, pages 3–12, april 2002. doi: 10.1109/ASYNC.2002.1000291.

[14] J. Cortadella, M. Kishinevsky, and B. Grundmann. Synthesis of synchronous elastic architectures. In *43rd ACM/IEEE Design Automation Conference (DAC 2006)*, pages 657–662, 0-0 2006. doi: 10.1109/DAC.2006.229277.

[15] William James Dally and Brian Towles. *Principles and practices of interconnection networks*. Morgan Kaufmann, 2004.

[16] Sergio Tota, Mario R. Casu, and Luca Macchiarulo. Implementation analysis of noc: a mpsoc trace-driven approach. In *Proceedings of the 16th ACM Great Lakes symposium on VLSI*, GLSVLSI '06, pages 204–209, New York, NY, USA, 2006. ACM. ISBN 1-59593-347-6. doi: 10.1145/1127908.1127957. URL http://doi.acm.org/10.1145/1127908.1127957.

[17] Q. Wu, P. Juang, M. Martonosi, L. S. Peh, and D. W. Clark. Formal control techniques for power-performance management. *IEEE Micro*, 25(5):52–62, Sept.-Oct. 2005.

[18] S.R. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, A. Singh, T. Jacob, S. Jain, V. Erraguntla, C. Roberts, Y. Hoskote, N. Borkar, and

S. Borkar. An 80-tile sub-100-w teraflops processor in. *IEEE Journal of Solid-State Circuits*, 43(1):29 –41, jan. 2008. ISSN 0018-9200. doi: 10.1109/JSSC.2007.910957.

[19] Andrea Bianco, Paolo Giaccone, Mario R. Casu, and Nanfang Li. Exploiting space diversity and dynamic voltage frequency scaling in multiplane network-on-chips. In *To appear in Proc. Globecom 2012*, december 2012.

[20] Xi Chen, Zheng Xu, Hyungjun Kim, P. Gratz, Jiang Hu, M. Kishinevsky, and U. Ogras. In-network monitoring and control policy for dvfs of cmp networks-on-chip and last level caches. In *Sixth IEEE/ACM International Symposium on Networks on Chip (NoCS)*, pages 43 –50, may 2012. doi: 10.1109/NOCS.2012.12.

[21] Manoj Kumar Yadav, Mario R Casu, and Maurizio Zamboni. Laura-noc: local automatic rate adjustment in network-on-chips with a simple dvfs. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 60(10):647–651, 2013.

[22] Liang Guang, E. Nigussie, and H. Tenhunen. Run-time communication bypassing for energy-efficient, low-latency per-core dvfs on network-on-chip. In *2010 IEEE International SOC Conference (SOCC)*, pages 481 –486, sept. 2010. doi: 10.1109/SOCC.2010.5784674.

[23] C. Seiculescu, S. Murali, L. Benini, and G. De Micheli. Comparative analysis of nocs for two-dimensional versus three-dimensional socs supporting multiple voltage and frequency islands. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(5):364 –368, may 2010. ISSN 1549-7747. doi: 10.1109/TCSII.2010.2047320.

[24] S.N. Wooters, B.H. Calhoun, and T.N. Blalock. An energy-efficient subthreshold level converter in 130-nm cmos. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 57(4):290 –294, april 2010. ISSN 1549-7747. doi: 10.1109/TCSII.2010.2043471.

[25] D. Bertozzi et al. Noc synthesis flow for customized domain specific multiprocessor systems-on-chip. *IEEE Trans. Parallel Distrib. Syst*, 16(2):113–129, 2005.

[26] U.Y. Ogras, R. Marculescu, and D. Marculescu. Variation-adaptive feedback control for networks-on-chip with multiple clock domains. In *45th ACM/IEEE Design Automation Conference, 2008. DAC 2008.*, pages 614 –619, june 2008.

[27] Seung Eun Lee and Nader Bagherzadeh. A variable frequency link for a power-aware network-on-chip (noc). *Integration, the VLSI Journal*, 42(4):479 – 485, 2009. ISSN 0167-9260. doi: 10.1016/j.vlsi.2009.01.002. URL http://www.sciencedirect.com/science/article/pii/S0167926009000030.

[28] Liang Guang, Ethiopia Nigussie, Lauri Koskinen, and Hannu Tenhunen. Autonomous dvfs on supply islands for energy-constrained noc communication. In *Architecture of Computing Systems – ARCS 2009*, volume 5455 of *Lecture Notes in Computer Science*, pages 183–194. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-00453-7. doi: 10.1007/978-3-642-00454-4_19.

[29] Abbas Rahimi, Mostafa E. Salehi, Siamak Mohammadi, and Sied Mehdi Fakhraie. Low-energy gals noc with fifo—monitoring dynamic voltage scaling. *Microelectronics Journal*, 42(6):889 – 896, 2011. ISSN 0026-2692.

[30] A.K. Mishra, R. Das, S. Eachempati, R. Iyer, N. Vijaykrishnan, and C.R. Das. A case for dynamic frequency tuning in on-chip networks. In *42nd Annual IEEE/ACM International Symposium on Microarchitecture, 2009. MICRO-42.*, pages 292 –303, dec. 2009.

[31] R. Bondade and Dongsheng Ma. Self-reconfigurable channel data buffering scheme and circuit design for adaptive flow control in power-efficient network-on-chips. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 57(11):2890 –2903, nov. 2010. ISSN 1549-8328. doi: 10.1109/TCSI.2010.2048773.

[32] L. Shang, L. S. Peh, and N. K. Jha. Dynamic voltage scaling with links for power optimization of interconnection networks. *HPCA*, pages 123–124, 2003.

[33] Q. Wu, P. Juang, M. Martonosi, and D. W. Clark. Formal online methods for voltage/frequency control in multiple clock domain microprocessors. *SIGARCH Comput. Archit. News*, 32(5):248–259, Oct 2004.

[34] Y. Shin et al. Power gating: Circuits, design methodologies, and best practice for standard-cell vlsi designs. *ACM Trans. Des. Autom. Electron. Syst.*, 15(4):28:1–28:37, Oct 2010.

[35] Xuning Chen and Li-Shiuan Peh. Leakage power modeling and optimization in interconnection networks. In *Proceedings of the 2003 international symposium on Low power electronics and design*, pages 90–95. ACM, 2003.

[36] Gwangsun Kim, John Kim, and Sungjoo Yoo. Flexibuffer: Reducing leakage power in on-chip network routers. In *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, pages 936–941. IEEE, 2011.

[37] Hiroki Matsutani, Michihiro Koibuchi, Daisuke Ikebuchi, Kimiyoshi Usami, Hiroshi Nakamura, and Hideharu Amano. Performance, area, and power evaluations of ultrafine-grained run-time power-gating routers for cmps. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 30(4):520–533, 2011.